UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Tècnica Superior d'Enginyeria
de Telecomunicació de Barcelona

ANALYSIS OF CLOCK TREE IMPLEMENTATION ON ASIC BLOCK QOR

A Master's Thesis

Submitted to the Faculty of the

Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

Universitat Politècnica de Catalunya

by

Javier Antúnez Sánchez

In partial fulfilment

of the requirements for the degree of

MASTER IN ELECTRONIC ENGINEERING

Advisor: Francesc Moll Echeto

Barcelona, October 2017

Title of the thesis: Analysis of clock tree implementation on ASIC block *QoR*

Author: Javier Antúnez Sánchez

Advisor: Francesc Moll Echeto

Abstract

The scope of this project is to develop a base methodology for clock tree synthesis that can improve the base results regarding the clock structure. The analysis of results will be done with a *Quality of Results* sets of metrics and by analysing the physical structure of the clock.

The analysis has been performed on three blocks with different physical characteristics to achieve a transversal solution. The initial tests performed have been focused on configuration options of the EDA tool used but were disregarded. The main tests upon this thesis is based are referred to the clock physical structure such as *fanout* constraints, *slew* constraints and clock cell selection.

One of the main results obtained is the importance of the layout of the block to set up the optimal constraints, limiting the transversal solution approach. It is as well an important point considering the internal algorithms followed by the tool.

Acknowledgements

I would like to acknowledge the help and knowledge provided by Ferran Martorell Cid and all the ASIC and Layout Design engineers at Esilicon for all the help granted for the development of this project, both in the project orientation as for help learning the methodology to be followed.

I would like to thank Esilicon for the chance of doing my master thesis project as it has been a great experience to go through.

I want to acknowledge too the help provided by Francesc Moll, to set up this project and by helping me whenever is necessary and helping me polish the project as well as structuring and giving format to this document.

Revision history and approval record

Revision	Date	Purpose
0	13/10/2017	Document creation
1	15/10/2017	Document revision - Draft 1
2	15/10/2017	Document expansion - Draft 2
3	16/10/2017	Document expansion - Draft 3
4	17/10/2017	Document revision and correction - Draft 4
5	19/10/2017	Final Version

Written by:		Reviewed and approved by:	
Date	12/10/2017	Date	19/10/2017
Name	Javier Antúnez Sánchez	Name	Francesc Moll
Position	Project Author	Position	Project Supervisor

Table of contents

List of Figures

List of Tables

1. Introduction

High frequency ASICs require large clock structures that toggle continuously. On the state-of-the-art chip clock tree power becomes one of the main contributors to the total power consumption.

Besides the direct consumption of power and area due to clock tree buffering there is also the indirect effect of clock *skew* on the number of hold buffers and timing closure effort.

Congestion problems can also be generated due to a bad clock tree building strategy.

The main goal of this project is to investigate ways to improve the clock tree building using EDA tools and quantify and qualify the results obtained and the direct and indirect effects of different clock tree building strategies using state-of-the-art blocks using sub-20nm *FinFET* technologies.

1.1. Requirements and Specifications

The scope of this master thesis is to generate a set of constraints to optimize the clock tree synthesis using EDA tools. To check the optimal constraints, they will be tested against several test blocks with different physical characteristics.

To assess the results obtained, it will be defined a set of metrics that will conform the *Quality of Results*. The *Quality of Results* will be used to determine from the metrics checked, which yields the best results and consequences on why it happens.

The first objective of the master thesis will be focused on how different configuration options provided by the EDA tool used on the test blocks can be used to improve the *Quality of Results* of all the blocks and try to obtain a transversal solution.

The second objective will be focused on the optimization of the clock structure by modifying constraints regarding their physical clock structure and design constraints.

1.2. Statement of purpose

The main objectives that are being covered on this master thesis are the following ones:

- Learning of the EDA tools used by the company where this master thesis is being developed.
- Analysis of state-of-the-art clock tree building techniques and methodologies regarding clock tree structures.
- Assessment of *Quality of Results* metrics used to analyse the results obtained on the different blocks being tested.
- Generation of experiment sets to be tested.
- Development of a test structure within the software to automatize the experiment testing in each block and the extraction of the metrics.
- Analysis of the metrics obtained by the tool and selection of the best experiment tests performed.

1.3. Methods and procedures

This project has been developed within *ICC2 Synopsys* EDA tools. The blocks upon the project has been developed have been provided by *Esilicon S.L.* such as the basic flow structure that runs on *ICC2*. Some additional scripts used have been designed by *Synopsys* engineers being used to extract some of the *Quality of Results* metrics.

All the scripts included on this thesis have been developed personally and focus on metric extraction. Some parts of the code have been inspired by codes provided by *Esilicon S.L.* and *Synopsys.*

1.4. Work plan

The project in a general approach can be separated in two main tasks.

- Learning of the development environment: The project was developed on the *ICC2* EDA tool. The first part of the project involved gaining some competency in how the software environment and the company flow works.
- Project development: The main tasks that have been developed regarding the main project are the analysis of the blocks, generation of the auxiliary scripts and result analysis.

In a more detailed list, the work plan in chronological order is the following one:

- Research on Clock Tree Synthesis:
  a. Research on clock tree synthesis and basic metrics regarding CTS.
  b. Research on methodologies focused on CTS improvement.

- Software learning.
  a. Basic courses around *ICC2.*
  b. Company and software environment learning through a learning block provided by the company through all the steps.
  c. Analysis of the different company scripts to identify the procedures followed on each step.
  d. Learning of *ICC2* and company environment through scripts and graphical interface.
  e. Analysis of the reports generated by the tool for future work.
- Milestone: Acquiring competence in the software and company environment to develop the project.

- Generation of basic auxiliary scripts regarding report generation:

  a. Selection of useful metrics of the data that can be provided by the tool and generated reports.
  b. Learning of *TCL* expressions regarding data parsing through tutorials and *Esilicon S.L.* and *Synopsys* scripts.
  c. Creation of the script that must call reports, obtain the relevant data and save it on separate files depending on what area they belong.

- Test of the blocks to be analysed:
  a. Unmodified flow running, script testing and initial metrics extraction.

- Generation of the initial experiment sets and results extraction:
  a. Analysis of the tool configuration option sets using the online documentation provided by *Synopsys.*
  b. Addition of the configuration option sets to the company flow structure.
  c. Running of the experiment sets on the test block and existing analysis blocks and extraction of initial results.
  d. Deprecation of the experiment sets due to faulty methodology.

- Generation of the test structure:
  a. Automatic run generation without using the graphical interface.
  b. Modification of the company flow to include several auxiliary scripts.

c. Modification of the report scripts to use a modified set of the *Quality of Results* Metrics.

- Generation of new experiment sets regarding physical modifications:
  a. Selection of the variations with help from company colleagues.
  b. Test of the configuration options in the blocks.
  c. Extraction of results, previous analysis and further experiment sets generations.

- Generation of a clock analysis script.
  a. Analysis of the report structure.
  b. Test of regular expressions to extract information.
  c. Basic script generation.
  d. Refinement of the script adding further functionalities.

- Final analysis of the experiment sets.
  a. Inclusion of final test block.
  b. Running missing experiment sets.
  c. Creation of a block combiner script.
  d. Obtaining *QoR* data and analysis.

- Thesis documentation

1.5. Gantt diagram

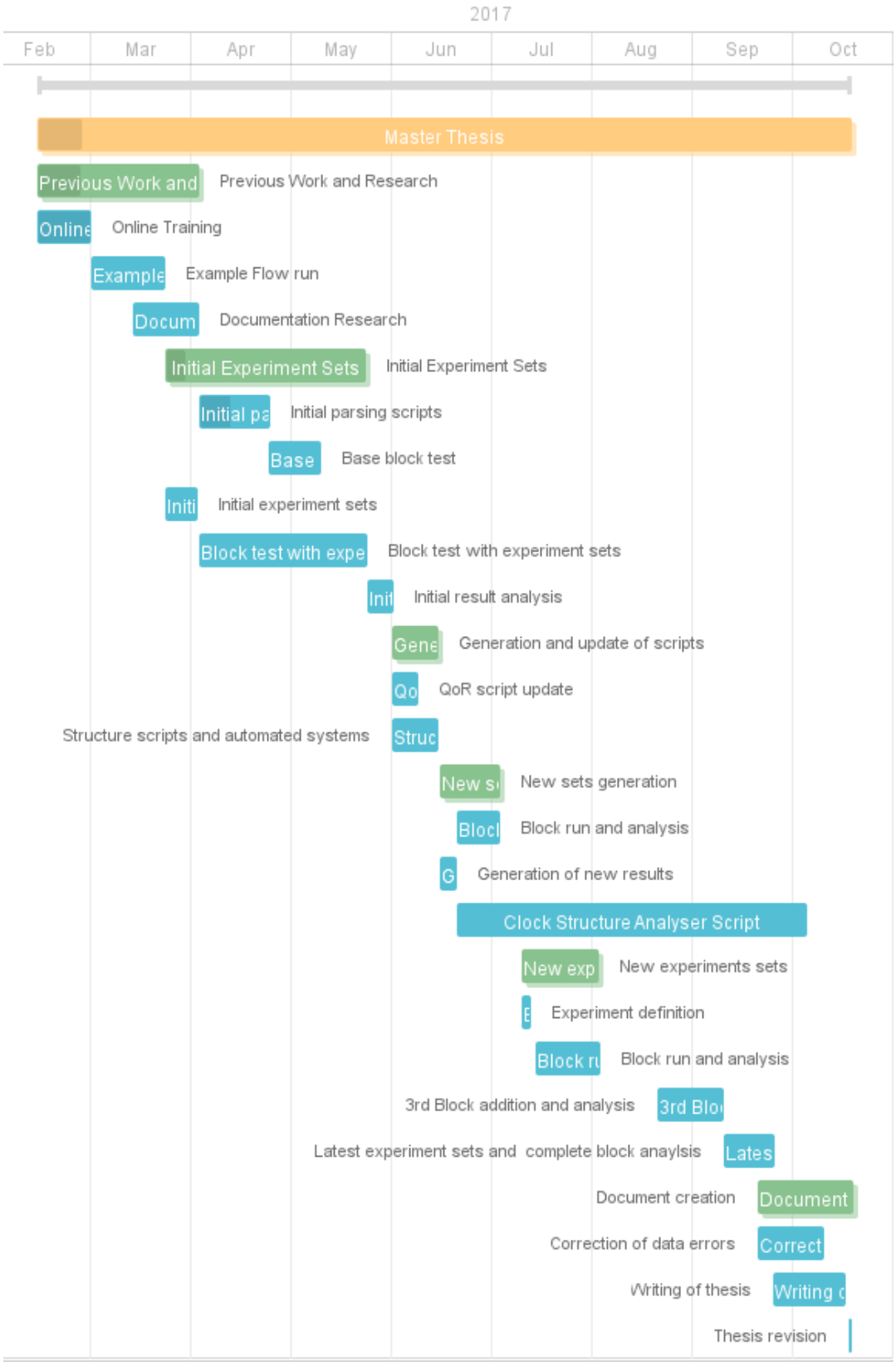

Figure 1.1: *Thesis Gantt diagram.*

2. State of the art of the technology used or applied in this thesis

The state-of-the-art will cover the current technology used in the project as well as information on clock tree synthesis techniques.

2.1. Planar CMOS to FinFET

The development of this Master Thesis is focused on the optimization of clock tree building in ASIC blocks. The technology on which all the blocks have been developed is sub-20nm *FinFET* and thus will be part of the state-of-the-art of this project.

2.1.1. Planar CMOS technology and its limitations

During a long time, planar CMOS technology has been the standard on IC design. In general terms it is reliable, cheap to design and it was possible to scale while improving the performance.

The Moore's Law, formulated in 1965 by Gordon Moore, predicted doubling on the number of transistors on integrated circuits every 18 months. This theorem explained the scaling of transistor's node technology up to 2010 approximately.

Planar CMOS technology has provided improvements in performance with scaling up to the 28 nm technology node. The main limitation for further scaling using planar CMOS is the increase of the leakage current as the technology nodes become smaller.

In general terms, the power consumption of a given integrated circuit can be expressed as:

$$P = \alpha \cdot C \cdot V_{DD}^2 \cdot f + V_{DD} \cdot I_{leakage} \quad (1)$$

The dynamic power depends of the circuit capacitance, the transistor capacitance, the number of gates, the power supply voltage, the frequency and the activity factor of the circuit. The activity factor of a given circuit depends on the probability of the integrated circuit working. This activity factor takes in consideration clock gating and cutting of power to certain parts of the circuit.

The usage of smaller technology nodes seeks mainly to increase the maximum frequency achievable, the optimization of power consumption.

The increase of the maximum frequency requires decreasing the transistor parasitic capacitance. One of the parameters to define the maximum frequency is the transition frequency. It is defined as the frequency on which the small signal gain of a transistor is 1.

It is defined as:

$$w_T = \frac{\sqrt{2 \cdot I_D \cdot \mu \cdot C_{ox} \cdot \frac{W}{L}}}{C_{gs}} \quad (2)$$

$$f_T = \frac{\sqrt{2 \cdot I_D \cdot \mu \cdot C_{ox} \cdot \frac{W}{L}}}{2 \cdot \pi \cdot C_{gs}} \quad (3)$$

Usually, the maximum frequency will be decades below the transition frequency, however it serves to determine an approximate maximum frequency. From expression (3), increasing the transition frequency are increasing the drain current, the aspect ratio of the transistors or reducing the gate capacitance of a transistor.

Assuming that the aspect ratio cannot be increased indefinitely and that increasing the drain current results on an increase of the power consumption, it is discarded.

Thus, the main option to increase the maximum frequency is reducing the gate capacitance. The gate capacitance depends on the transistor size, thus transistor scaling reduces the parasitic capacitance and increases the maximum achievable frequency.

The reduction on the transistor size requires lowering the supply voltage to keep in check the electric field of the transistor. With the reduction of the power supply, it is needed a reduction on the threshold voltage to keep a correct switching operation.

Thus in general terms, from expression (1), the dynamic power consumption is kept on check with the reduction of the power supply despite the frequency increase, assuming a constant activity factor and capacitance.

Regarding the leakage power, it depends mainly on two factors, the circuit power supply and the leakage power. As seen before, technology scaling requires a reduction on the power supply voltage to keep a stable electric field.

The leakage current or subthreshold current on a transistor can be defined as:

$$I_{sub} \propto I_o \cdot e^{\frac{-V_T}{n \cdot K \cdot T/q}} \propto I_o \cdot e^{\frac{-V_T}{n \cdot u_T}} \quad (4)$$

$$I_{sub} = I_s \cdot e^{q \cdot \frac{V_{GS} - V_T}{n \cdot K \cdot T}} \cdot \left(1 - e^{-q \cdot \frac{V_{DS}}{KT}}\right) \quad (5)$$

Assuming I_o being approximately constant and knowing that the thermal voltage at a given temperature will also be constant, the subthreshold current will depend on the threshold voltage. Given that it is required to reduce the threshold voltage to ensure a correct switching operation, this results on an increase of the subthreshold current as the threshold voltage is reduced.

Then, considering the leakage power expression in (1), technology node scaling causes an exponential increase on the subthreshold voltage and a linear decrease on the power supply.

Moreover, the reduction on the technology node and thus the channel length, makes it impossible to achieve a completely off-state while keeping a good on-state current drive.

Usually, the maximum frequency will be decades below the transition frequency, however it serves to determine an approximate maximum frequency. From expression (3), increasing the transition frequency are increasing the drain current, the aspect ratio of the transistors or reducing the gate capacitance of a transistor.

Assuming that the aspect ratio cannot be increased indefinitely and that increasing the drain current results on an increase of the power consumption, it is discarded.

Thus, the main option to increase the maximum frequency is reducing the gate capacitance. The gate capacitance depends on the transistor size, thus transistor scaling reduces the parasitic capacitance and increases the maximum achievable frequency.

The reduction on the transistor size requires lowering the supply voltage to keep in check the electric field of the transistor. With the reduction of the power supply, it is needed a reduction on the threshold voltage to keep a correct switching operation.

Thus in general terms, from expression (1), the power dynamic power consumption is kept on check with the reduction of the power supply despite the frequency increase, assuming a constant activity factor and capacitance.

Assuming I_o being approximately constant and knowing that the thermal voltage at a given temperature will also be constant, the subthreshold current will depend on the threshold voltage. Given that it is required to reduce the threshold voltage to ensure a correct switching operation, this results on an increase of the subthreshold current as the threshold voltage is reduced.

Then, considering the leakage power expression in (1), technology node scaling causes an exponential increase on the subthreshold voltage and a linear decrease on the power supply.

Moreover, the reduction on the technology node and thus the channel length, makes it impossible to achieve a completely off-state while keeping a good on-state current drive.

2.1.2. FinFET technology and its advantages

As specified before, the technology used is sub-28nm *FinFET*. On this section, it will be discussed the advantages and limitations of *FinFET*.

It has been seen that with newer technology nodes, the leakage current and thus the leakage power increases being comparable to the dynamic power consumption on a given integrated circuit.

FinFET technology uses a tri-dimensional gate, also called fin. *FinFET*s can have mainly two structures, the double-gate structure and the tri-gate structure.

The double-gate structure has the drain and source connected by the fin. The gate of the device is placed at both sides of the fin and covers with a ultra-thin layer of silicon the fin to connect the gate at both sides.

On the other hand, on tri-gate structures, the gate covers completely the fin, allowing gate control from either the sides or top of the fin.

One of the main advantages of the *FinFET* over planar technologies is the capability of increasing the width of the channel in a much easier way. Considering a given fin transistor, the effective width of the channel for a tri-gate structure will be:

$$W_{eff} = 2 \cdot H_{fin} + W_{fin} \quad (6)$$

Where H_{fin} is the fin height and W_{fin} is the fin width.

The physical structure of a *FinFET* is:

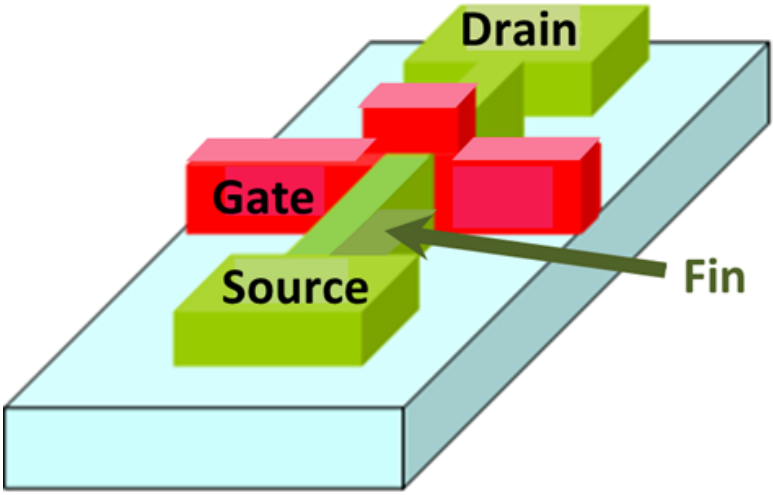

Figure 2.1: *Physical structure of a FinFET. [8]*

Knowing that the current is dependent on the width of the channel, it is possible to increase the driving current of a given transistor by increasing the fin height. Moreover, this tri-dimensional structure, makes possible to achieve higher levels of transistor density by increasing the fin height instead of making wider transistors, when comparing to planar CMOS.

Given the nature of the gate in tri-gate structures, it is easier to control the transistor in smaller technology nodes, as opposed to planar CMOS where for a similar technology node, the required threshold voltage would difficult a proper switching.

This results also on a reduced leakage current with a lower threshold voltage due to better gate control. The use of the fin also makes redundant the doping of the channel in order to prevent Drain Induced Barrier Lowering and other short-channel effects that appeared in planar CMOS technology.

The reduction on the threshold voltage allows as well the reduction of the power supply voltage, effectively lowering the dynamic and leakage power of a transistor.

The elimination of channel doping processes increases carrier mobility and reduces the process variability.

On the other hand, the use of a tri-dimensional structure poses many challenges that have to be solved.

There is less flexibility in the control of the on-state current drive due to the nature of the fin. The effective channel width will depend on the fin height and on the number of fin fingers used on the design.

Multiple fins are used in order to limit the drain-source resistance introduced by reducing the fin width. By using multiple fins, the total channel width can be estimated to:

$$W_{tot} = W_{eff} \cdot N_{fin} = (2 \cdot H_{fin} + W_{fin}) \cdot N_{fin} \approx 2 \cdot H_{fin} \cdot N_{fin} \quad (7)$$

This expression considers that the fin width is negligible compared to the fin height. The fin height will be fixed by the technology considering technical limitations and process design.

As it can be seen, the channel width and thus the driving strength becomes quantized and can only be increased by the number of fins, compared to planar CMOS where it is possible to achieve better control the on-state current by modifying the channel width.

It must also be considered tri-dimensional structure for physical layout design. It must be considered in closer detail the placement of transistors. Previously, for the designer it was not as important to know the relationship between the design layout and the manufacturing process.

This however cannot be applied to *FinFET* structures. It is necessary by the technology provider to give detailed information on the physical characteristics of the device and it is necessary for the designer to have knowledge on the manufacturing process.

2.2. Terminology and Concepts

The main topic of this thesis is the optimization of the clock structure on ASIC blocks. In this section, it will be covered the main definitions regarding clock power.

The *Worst Negative Slack* is referred to the maximum difference between the clock and data arrival at given flip-flop. If the *Worst Negative Slack* obtained is lower than zero, it means that this path and the design is not meeting timing enclosure and thus will not work at the desired frequency.

The *Total Negative Slack* is the sum of all the slack values obtained on a given design.

The *Skew* indicates the timing difference of the earliest clock signal arrival and the latest clock signal arrival at the flip-flops of a design. The *skew* values obtained can be misleading and thus the term *Local skew* is used.

The *Local skew*, in contrast with the Global *Skew* indicates the maximum timing difference of the clock arrival at flip-flops of a given timing path. *Local skew* will then not consider the maximum timing difference between flip-flops that do not share the same timing path and will give more accurate timing information.

The *Latency* or *Insertion delay*, indicates which is the time it takes for the clock signal to propagate from the clock source to the furthest flip-flop of the design.

The Utilization of a given design is the quotient between the area used by standard cells and macro cells and the total area of a design. The tool used defines the utilization as the quotient between the standard cell area and the total area.

2.3. Adjustable Delay Buffers

The next sections will cover the use of several techniques referring to clock tree synthesis and improvement on the clock tree structure. The first section will cover the use of Adjustable Delay Buffers, while the second one will cover the use of reconfigurable clock trees.

On current designs, the effect of process variations such as temperature, voltage on multi-power mode designs can affect and degrade the performance of a given block. All this processes variations and the problems they may cause, become much more severe as technology scales down.

One of the main problems to arise is the hold and setup timing violations. The use of hold-fixing and effective *skew* management techniques can be used in order to improve results. Several other methodologies have been explored to deal with process variation such as buffer insertion and sizing and wire sizing.

However on multi-power designs with on-chip-variations considered, the resulting clock can present high wirelengths that may affect negatively signal routing or have no clock distribution that meets the timing requirements of the design on all the scenarios.

Adjustable Delay Buffers (ADBs) are used on post-silicon tuning to deal with timing enclosure. ADBs can have their delay modified by a control input.

By replacing some buffers on the base design by ADBs it can be possible to modify the delay they present for different power modes or OCV. However, ADBs in comparison to regular buffers they require a bigger area, additional control logic and have a higher power consumption. This sets a limit on how many buffers can be inserted to avoid excessive power consumption.

The problem for buffer insertion can be formulated as:

- Add the minimum number of buffers that allow meeting mode *skew* requirements with the least power overhead.
- Meet the global *skew* requirements imposed by the design.

This first algorithm for ADB insertion takes into consideration the global design *skew* or local module *skew*. Under these considerations, although the *skew* requirements are met, there can be hold and setup time violations on individual nodes across multiple power modes and domains.

Considering the setup and *skew* bounds as:

$$x_j - x_i \geq D_{max}(Ckt_{i,j}) + t_{setup} - t_{clk} \quad (8)$$

$$x_i - x_j \geq t_{hold} - D_{min}(Ckt_{i,j}) \quad (9)$$

x_i and x_j are the clock arrival at sinks i and j (where data goes from i to j). $D_{max}(Ckt_{i,j})$ and $D_{min}(Ckt_{i,j})$ are the maximum and minimum delays from the output of sink i to the input of sink j. t_{clk} is the clock period. t_{setup} and t_{hold} are the setup and hold constraints that must be met. If the inequalities are met, then the design will have no setup or hold violations.

The addition of ADBs will allow modifying the delay between the sinks and allowing meeting the timing constraints.

Negative setup time violations are quite hard to fix as it will usually imply increasing the clock period and thus lowering the maximum frequency of the circuit. It can also be fixed by optimizing the datapath structure.

However, negative hold time violations imply that data can change before the clock captures it. This can be fixed easier by adding delays to the datapath in order to delay the data arrival.

By modifying existing ADB insertion algorithms to take into consideration setup and hold time, it is possible to achieve designs that can meet timing enclosure as well as reducing the number of violations of the design.

As explained before, as technology scales down, on-chip-variation effects become more prevalent. This makes timing enclosure harder and thus worsens chip performance. Frequency scaling has also limited the *skew* bounds to meet timing enclosure.

Process variations considered to be relevant to affect buffer delay are wire width, supply voltage, temperature, load capacitance and input *slew*.

2.4. Reconfigurable Clock Trees for Multi-Corner Multi-Mode designs

Clock network of integrated circuits must be able to operate on multiple corners and multiple modes (MCMM).

It is purposed the use of multiple clock trees to meet the *skew* requirements for each scenarios. As building several clock trees it requires an increase of area and power consumption, the bottom end of the tree is shared and always active.

To determine which parts of the clock network are active, or gates and a one-input n-outputs demultiplexer is used.

Each scenario will have different *skew* constraints. The *skew* for each scenario will take into consideration a safety margin.

If two subtrees are active on a group of scenarios, a feasible *skew* graph will be made. This graph will contain a paired table of all sinks and *skew* constraints.

After applying the safety margin on the *skew* constraints, both subtrees will be paired and if a common merging region exists (*skew* margin for the common sinks ≥ 0), both subtrees can be merged obtaining a feasible clock schedule for the new tree.

If the merging region of both subtrees is empty (the *skew* margin for common sinks < 0).

Two type of incompatibility are considered, mode and scenario incompatibility. When two subtrees are scenario incompatible but mode compatible it implies that there is no common merging region. If by decreasing the safety margin applied initially a common merging region exists then the two subtrees are scenario compatible and can be merged.

Mode incompatible trees will not be able to be merged in any case and will be buffered and controlled by combinational logic on the final tree.

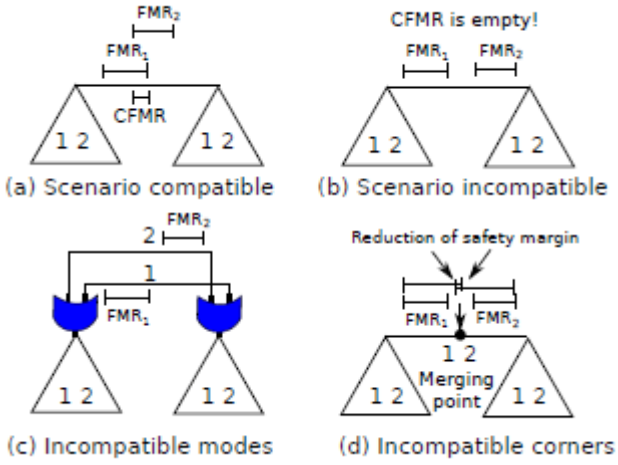

Figure 2.2: *Clock tree merging possibilities depending on scenario and mode incompatibilities.*

3. Methodology and Project Development

The Master thesis was initially focused on analysing how the modifications of the configuration options on the EDA tool used could improve the results obtained regarding clock building.

Mid-project, that approach was abandoned due to the reasons presented in the next chapter and a new focus was followed.

The new approach upon which the majority of the master thesis will be focused is regarding clock tree structural changes such as the *slew* or the *fanout* constraint.

This chapter will cover how the project structure has been developed in order to generate an automated block and report system.

It will also be presented the blocks being used on this thesis as well as which experiment sets have been used and the reasoning behind them while the most relevant results will be presented on the next chapter.

3.1. Block Definition and Characteristics

To test the different experiment sets that will be explained later it is necessary to select different blocks to present a transversal picture. To make an appropriate block constellation several conditions should be met:

- Same technology in all blocks: To have more reliable results due to the limitations on the tests developed all blocks should be developed on the same technology to reduce variability in the results and to ease analysis.
- Blocks with different characteristics: The blocks should be different enough and have different characteristics in terms of physical layout.

By having different blocks it might be possible to give an optimal set of configurations for all blocks.

Block selection will be done considering their layout distribution. Three blocks have been selected: a block dominated by memory, a block dominated by logic and a mixed block.

The usage of memories affects how flip-flop distribution and routing is done. Most usually, memory placement in a block requires of a restricted area around it where neither cells nor routing can be done.

This differs from mostly logical blocks where this restriction does not exist usually resulting on a better and easier block development.

Asides from this block classification it will be considered the following characteristics to define each bloc:

- Number of flip-flops
- Number of memory instances
- Number of logic instances
- Total area
- Memory area and memory area percentage
- Logic area and logic area percentage
- Memory to logic ratio

3.1.1. Block 1

This block is memory dominated and has the following characteristics:

Number of flip-flops	433,290
Number of memory instances	417
Number of logic instances	329177
Total area	5,924,460.81 um2
Utilization	0.3186
Memory area	3,329,044.77 um2
Memory area percentage	56.19%
Logic area	792194.14 um2
Logic area percentage	13.37%
Memory to logic ratio	4.2

Table 3.1: Block 1 characteristics.

In terms of area and physical layout this is the most complex block. It has the most memory instances taking roughly 60% of the total area with a memory to logic ratio of 4.2.

In the physical layout the memories are placed all across the block. Flip-flops are distributed across all the block and in-between the memories. Open areas without memory blockages are not fully used to place flip-flops.

The following figure shows the memory placement and flop distribution of Block 1. The clock input pin is marked on green.

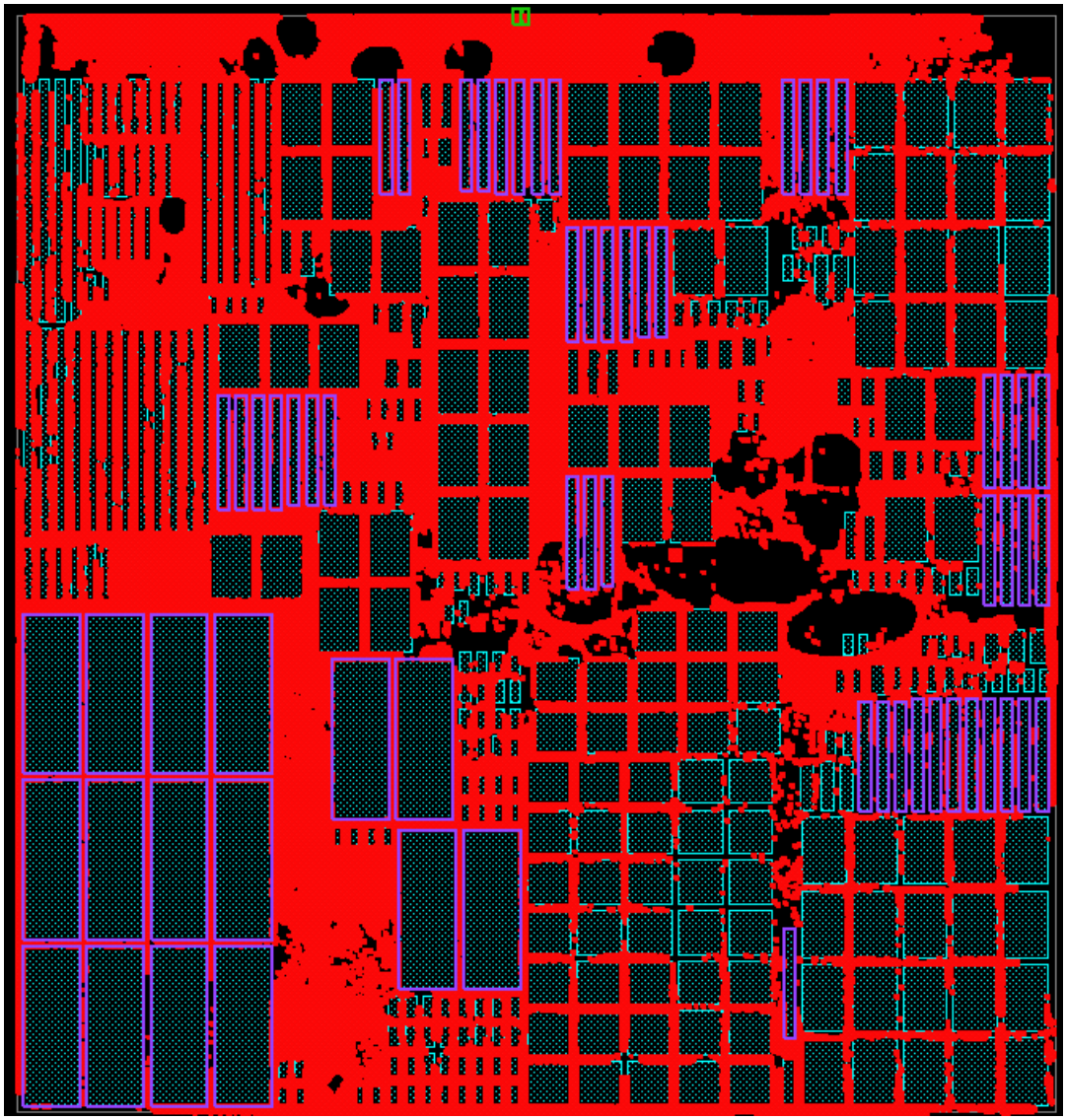

Figure 3.1: *Memory and flip-flop distribution at Block 1.*

3.1.2. Block 2

This block is considered as mixed and its characteristics are the following ones:

Number of flip-flops	347,531
Number of memory instances	110
Number of logic instances	243,314
Total area	2,132,247.05 um2
Utilization	0.4576
Memory area	856,478.89 um2
Memory area percentage	40.16%
Logic area	570,155.27 um2
Logic area percentage	26.73%
Memory to logic ratio	1.5

Table 3.2: Block 2 characteristics.

This block is considered as a mixed block in terms of memory and logic distribution as it has a memory to logic ratio of 4.2 compared to Block 1.

The physical layout on this case is different from Block 1. This block has less memory instances which take less area than on the first case. There are open spaces with no memory blockages that are used for flip-flop placement. The number of in-between memories flip-flops is also lower than on the first block.

The following figure shows the memory placement and flop distribution of Block 2. The clock input pin is marked on green.

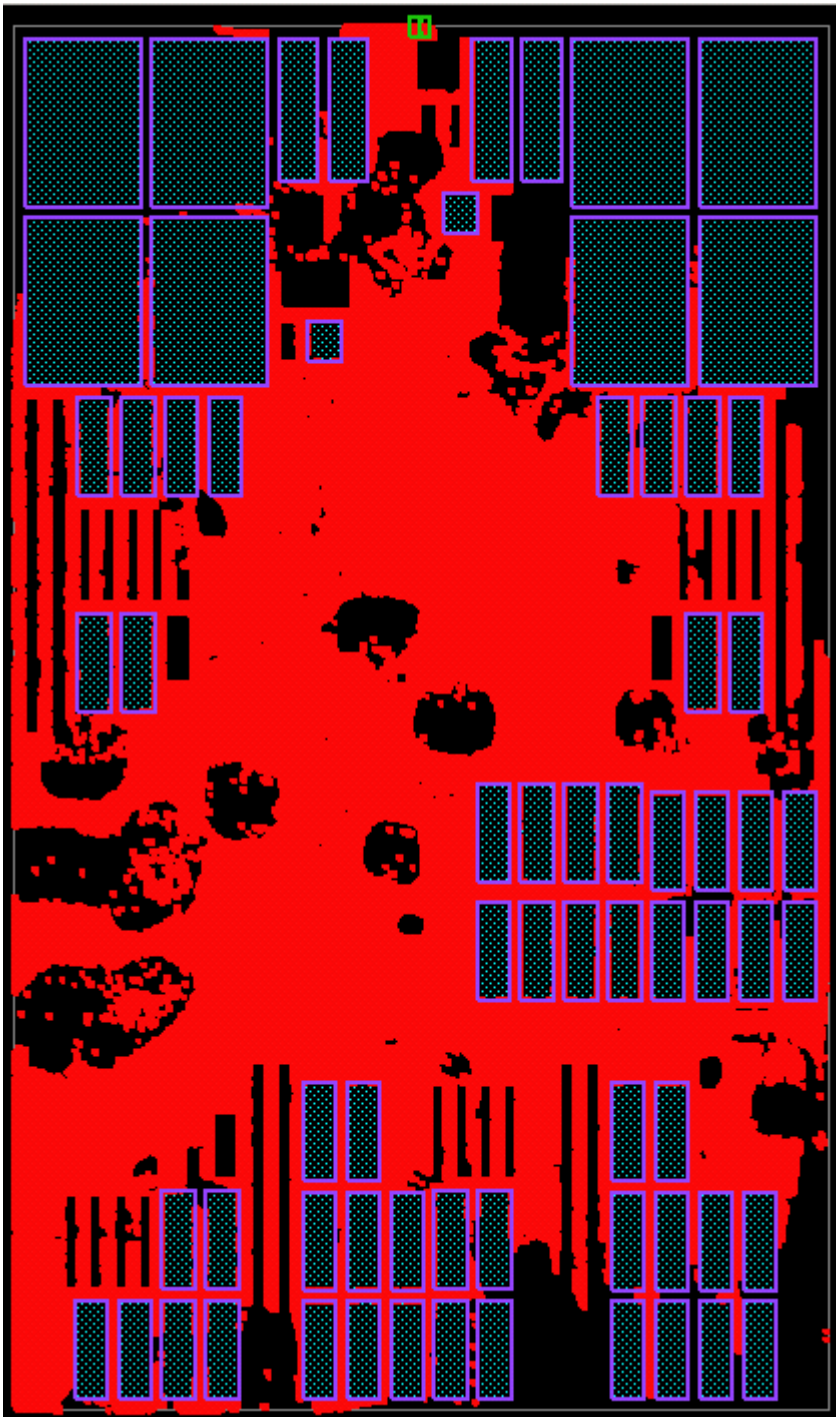

Figure 3.2: *Memory and flip-flop distribution at Block 2.*

3.1.3. Block 3

The final block considered to perform the different experiment sets is purely logic. Its characteristics are the following ones:

Number of flip-flops	339,959
Number of memory instances	0
Number of logic instances	2,084,106
Total area	1,276,547.46 um2
Utilization	0.3609
Memory area	0 um2
Memory area percentage	0
Logic area	460,769.45 um2
Logic area percentage	36.09%
Memory to logic ratio	0

Table 3.3: Block 3 characteristics.

On this case, there are no restrictions for flip-flop placement and routing as there are no memory or other types of blockages in the block layout.

The following figure shows the clock pin placement as well as the flip-flop placement across the block.

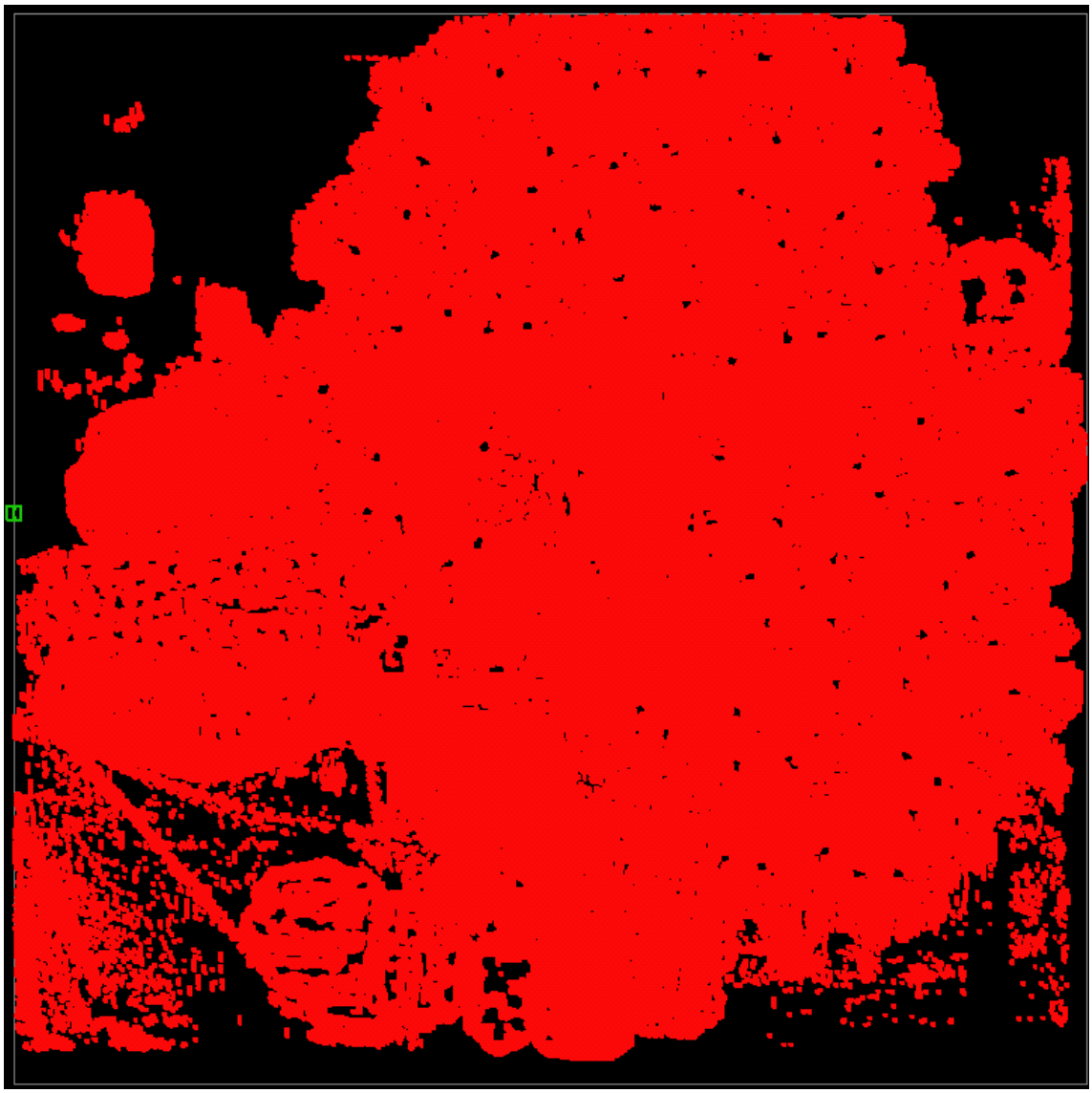

Figure 3.3: *Memory and flip-flop distribution at Block 3.*

3.1.4. Block 4

The final block does not classify into the test blocks used to develop this project. The objective of this block was to make an initial test in the configuration options and variations tested in the experiment blocks before passing them to the test blocks.

This block has a much simpler structure than the other blocks which allows having faster runtimes to correct possible errors that may happen in the sets test.

The block characteristics are the following ones:

Number of flip-flops	39,785
Number of memory instances	11
Number of logic instances	284,307
Total area	194,977.09 um2
Utilization	0.4528
Memory area	66,240.06 um2
Memory area percentage	33.97%
Logic area	63,370.27 um2
Logic area percentage	32.5%
Memory to logic ratio	1.05

Table 3.4: Block 4 characteristics.

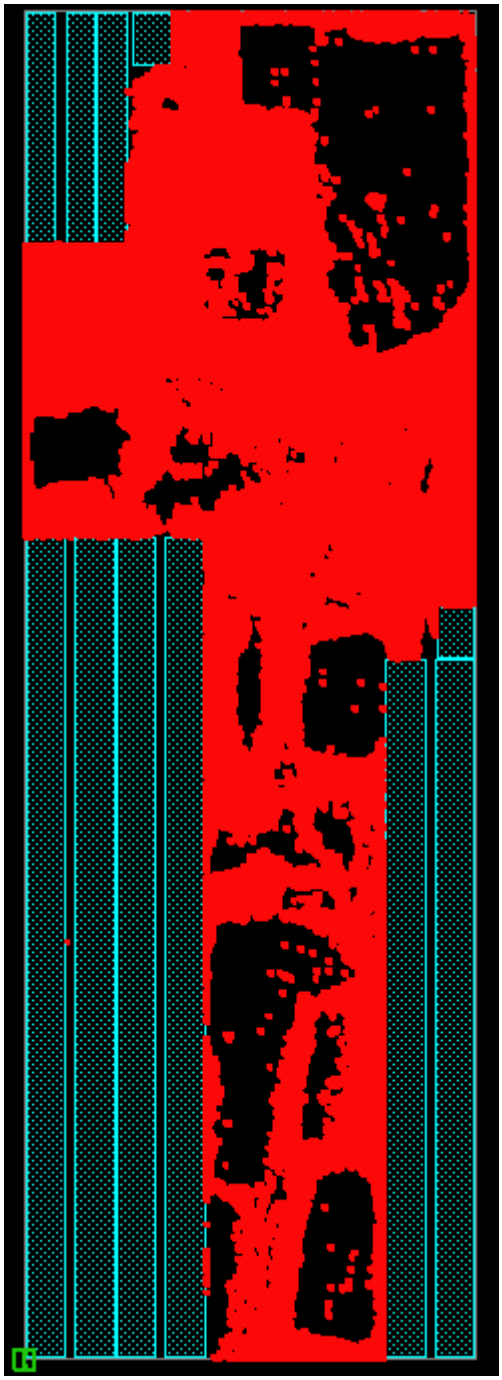

Figure 3.4: *Memory and flip-flop distribution at Block 4.*

3.2. Metrics Definition

To perform the analysis on the different experiment sets that have been performed, a group of metrics have been used to define the *Quality of Results*.

The definition of the metrics have been classified under two groups.

- Power Performance Analysis metrics (*PPA*): These are the main metrics used to analyse the results obtained, they are related to timing, clock power and clock area metrics.
- Support metrics: These metrics are used to support the results obtained and include *skew*, *insertion delay*, number of clock instances, utilization, *DRC* violations and clock structure.

3.2.1. Power Performance Analysis metrics

This metrics specified are the most important considered . The *Worst Negative Slack* is the main timing metric considered as it indicates if the block can work at the specified frequency.

The clock power and area give a general idea on the optimization of the clock structure. The clock power is relevant as the dynamic clock power can represent large portions of the total power.

The clock area, albeit it is not relevant when considering to the total area of the circuit can be used to compare different runs.

The number of clock instances can be used together with the clock power and area to compare between different runs.

3.2.2. Support metrics

The support metrics are used together with the PPA metrics.

The *skew* gives an indication of the clock tree balance and indicates the maximum difference of the clock arrival at the flip-flops of a block.

The *latency* is the maximum delay from the clock input pin to any of the end-points of the block.

The number of clock instances can be used together with the clock power and area to compare between different runs.

The utilization determines the total area used by the logic compared to the total block area.

The number of *DRC* violations can be used to indicate possible problems on a block. They indicate problems during the design in terms of spacing, overlapping, etc.

3.3. Scripts and Automatic Block Generation

This section will cover how the block and report generation system has been build up. The first part of this section will cover how the system in charge of generating blocks and the modifications to be made, as well as running the block and extracting the information of interest has been designed.

The second part of the section will cover the most relevant points of the scripts done in order to have this system.

In general lines, the system had to be able to use the basic shell scripts provided by the company that form the flow to be followed, while being flexible enough to introduce modifications and report the desired values.

3.3.1. Automatic Block generation

The automatic block generation system is made out of several sub-blocks. Essentially the system can be simplified in the following blocks:

- Reference shell blocks: Contains the reference blocks modified to fit the automated system.
- Configuration sets directory: Folder that contains all the configuration scripts in Tool Command Language *(TCL)*
- Auxiliary scripts directory: Folder that contains reporting scripts and other modification scripts for some options in Tool Command Language *(TCL)*
- General block directory: Directory in which all blocks are run. Has several subdirectories for each block type.
- Block generation script: Script in charge of copying the shell blocks, the auxiliary scripts and chosen configuration sets in the main block directory and start the flow.

The following figure summarizes the block structure used:

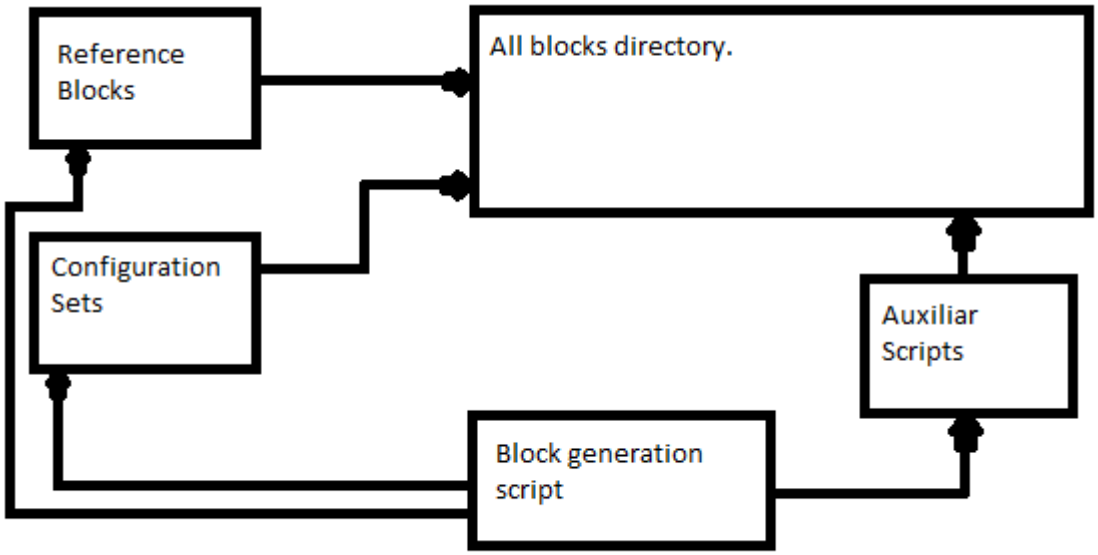

Figure 3.5: *Automatic Block Generation Structure.*

3.3.2. Reference shell blocks

The shell blocks contain the basic company flow, the constraints file and the floorplan file. This file has the following modifications:

- Activation of all the scenarios at each step to avoid having any scenarios inactive. This is necessary to obtain the best results across all scenarios by analysing the worst case.
- Inclusion of the configuration sets in all steps. This is done as some configuration options must be used first at different steps. By running in all steps this possible error is avoided.
- At the final of the routing stage, runs all the reporting scripts and stores them in the desired folders.

3.3.3. Configuration sets directory

As specified before, this directory contains all the desired configuration scripts. The script specified at block generation script is copied and added into a selected folder on the modified reference block.

Asides from some specified experiment sets that do require additional modification, the configuration script will be the only one needed to generate the desired modification.

3.3.4. Auxiliary scripts directory

This directory contains additional reports to add functionalities to the block generation script as well as other scripts that must be used. Most scripts are added automatically to the flow while some others are used manually for specific uses.

These auxiliary scripts can be classified on the following types:

- Reporter and parser scripts: Used to generate partial and final files that contain the desired reported information for the *Quality of Results*
- Clock Structure Analyser: Extracts information about the clock structure and stores it in a file.
- Clock pin modification script: This code changes the positioning of the clock pin placement for an experiment set.
- Report combiner: This script combines the *Quality of Results* obtained by the parser scripts and adds some mathematical operations to have more complete comparison results.

3.3.5. General block directory:

All the block runs are placed in the same directory folder. For each block a new subdirectory is needed. In each subdirectory the block flows are placed. The flows are based on the reference company flow with some modifications done to fit the designed system.

To ensure proper working it is needed to create the directory either manually or set it up in the block generation generation script.

3.3.6. Block generation script

All the following scripts and directories are managed by the block generation script. The block generation script follows this sequence.

- Creates block sub-directory in the general block directory.
- Copies the floorplan, scripts and auxiliary override folder.
- Copies the configuration script to the override folder and renames it.
- Copies the rest of auxiliary scripts.
- Changes to the subfolder and runs the start-up script from the copied flow.

Despite being able to automate the running of scripts flows it has some limitations that must be taken into consideration else the script will fail at the start-up script or give an erroneous block.

- The shell block flows must be modified to take into account the scripts added that must be run.
- The copied scripts must be placed at the same subdirectory within each block.
- In the block generator script, if any block should be added, the script must be added manually, that is, it has no self-modification capabilities to include newer blocks.

Within the specified parameters, the script behaves correctly while being quite inflexible if the block structure or script placement is out of the planned norm.

3.4. Script Analysis and Explanation

This section will include more detailed explanations and remarks on the scripts that have been used in the block generation script and other auxiliary scripts. Only the user-written scripts will be explained and included on the annexes while giving general explanations on company and *Synopsys* scripts.

3.4.1. Block Generation Script: *run_gen.tcl*

As explained on the previous section, this script is in charge of copying the desired scripts and running the set-up flow script to start and run the block.

As specified before, the initial code block of the script will copy and rename the needed scripts in the block folder.

After copying the desired scripts, sets up the main setup script of the flow, choosing different memory assignation depending on the block type.

One of the main points to take into consideration from the setup script calling:

- Typical setups will call an interface and a Graphical User Interface (GUI) while this script does not. This adds flexibility and reduces user managed steps, however it removes tolerance to errors as a block error will shut down the current job and give no error warning unless specified.
- User modifications are needed to change the memory requirements, log reports, type of machine used amongst others.

3.4.2. Reporter and parser: *report_parser.tcl*

This is the script in charge of generating reports and parsing them in a more compact format. The output files of this script are given in a Comma Separated Values format (.csv) to use them in other programs in order to analyse results.

The script has been designed to generate individual reports on different areas of interest such as:

- Total power information.
- Clock power information.
- *Insertion delay*.
- Routing information.
- *Local skew* reports.
- Wirelength information
- Layer congestion.
- Utilization.
- Clock area information.

By running the report manually it is also possible to execute this script on any step by selecting it in a script option.

Asides from the reports generation there are options to generate a more compact file with the final selected *QoR* file format. To avoid problems regarding scenario name due it being able to vary, the information is passed to the script manually.

The basic structure that is followed at each individual report and parsing of it is the following one:

- Check existence of scenarios.
- Generates the report or uses an existing report if present in the flow.
- Line by line analysis using the regular expression structure provided in the *TCL* package.
- Assignation to user made variables of the relevant information obtained via parsing.

This procedure is repeated for all individual report types, parsing and storing the information into user defined variables.

Once all the variables are obtained, individual *CSV* files are generated with all the information. Asides from the individual reports, a final file is generated containing the most relevant information. The information preparation and storage in a file is done as it follows:

- First, it is checked if the file must be generated and pre-preparation of obtained information is done.
- Once the data preparation is done, the file is created and opened. Then the information is written and the file is closed to avoid errors. It is necessary to define the separator used, to then make it available data sheet programs.

Most scripts with certain complexity in *TCL* require argument declaration to work correctly. The argument declaration is done outside of the main script and it must be parsed at the beginning of the script to extract the information.

At the argument declaration it must be declared how it is called, a description, and which type of variable it requires and whether it is optional or not.

3.4.3. Clock Structure Analyser: *clock_structure_analyser.tcl*

This script is in charge of analysing the physical clock structure file of a given block and extract relevant information. The main information being obtained is:

- Number of *repeater*s, Integrated Clock Gating Cells (*ICG)*, clock sinks, balance pins and clock sources.
- *Fanout* of *repeater*s and *ICGs*.
- Types of *repeater*s and *ICGs* used.
- Location of clock cells and sink pins.
- Wire and cell capacitance.

Asides from this information, this script provides basic math support to extract average *fanout*, average Manhattan distance between clock cells and sinks, average cell and wire capacitance, etc.

Debugging code has been added to check if the assumptions made in terms of possible line structures in the report exist.

Overall, the data parsing and handling has been done similarly to the *report_parser.tcl* script with some modifications to data treatment, focusing more on the use of arrays when possible as they prove to be better optimised and given the sheer size of the clock reports analysed can cut some machine time needed.

As done previously, the first step is checking if a clock structure report file exists, if it does not, or it is wanted anyways, a new file will be created on a user selected directory while keeping old files.

When the file is extracted, several counters and auxiliary variables will be declared to be used later. The counters will be used to obtain numbers on how many *repeater*s, *ICGs,* sink pins and balance pins are obtained. The sink pins are the flip-flops of the design while the balance pins are used to balance capacitance in the clock branches.

Five different structures have been found to be present in any clock structure file generated by the tool reporter and thus taken into consideration. There is one line type for clock sources, *repeater*s, *ICGs,* sink pins and balance pins.

For each type, it is checked if the current line follows its structure and then the information is extracted following the same pattern used in the other scripts that need parsing.

The obtention of the average and maximum repater and *ICG* is done iterating over the obtained arrays in the regular expression. For each position on the array, the *fanout* obtained is read and several counters are increased depending on the value of the *fanout*.

The Manhattan distance is defined as the distance measured along the defined axis. On this case, Cartesian axis are used and the total Manhattan distance will be defined as:

$$M_{distance} = x_{distance} + y_{distance} \quad (12)$$

On this case, the Manhattan distance has been calculated between and endpoint driver and the downstream sinks and balance pins.

The capacitance information is also obtained via the attribute of the nets connected to an endpoint driver.

3.4.4. Clock pin modification script: *replace_clockpin.tcl*

In some experiment sets, the input clockpin is centered and a superior layer is used to check if better results can be obtained in terms of clock building. The pin movement was done after the initial cell placement. To move the clock pin it is needed to give the initial clock name to locate and remove it, and then create a new clockpin on the new position.

To perform the pin movement several information is needed:

- Current clockpin name.
- Block dimensions.
- Pin shape type
- Pin margin.

The block dimensions are used to calculate the middle point of the block, while the pin margin is used to avoid conflict with other block tracks, memories, etc. The shape type is defined in by the tool and is needed to create the shape.

To move the clockpin, the existing shape must be deleted and a new clockshape is created.

3.4.5. Report combiner: *block_combiner.tcl*

Given how the block generator and report generation is done, one report is generated for each block. Because of that reason, it is needed an auxiliary report to combine existing report *CSV* files.

This script combines the existing reports given separated by block type and block name and has the capability of performing some mathematical operations to simplify the analysis. Similar to other blocks, the *CSV* file generation is done following a similar procedure as in other cases.

The block selection is done with an empty list that adds all the blocks on which data has been defined. Similarly to other cases, the existing data has been parsed in order to simplify the results.

For each list, the data is obtained using a regular expression as in previous cases. For each block, the report parsed file is analysed which simplifies obtaining the data.

A math flag has been enabled to perform some percentages to ease further analysis.

Once the data is obtained, if the math flag is active, it will perform some percentages to ease further analysis, the first block passed is used as a reference and the information obtained is passed to a new set of variables.

3.5. Methodology and Experiment Sets

As explained before, the methodology followed in the thesis development and result analysis has followed two main approaches. One followed on the selection of optimal configuration options within the EDA tool and one focused on physical modifications in the clock characteristics. On this section, all the experiment sets will be defined with explanations on why they were chosen.

3.5.1. Reference block parameters

The reference parameters used on each block are the following ones:

Default *Slew* Constraint	80 ps
Default Clock *Fanout* Constraint	32
Default *Repeater* Inverters	*INV_D16, INV_D12* and *INV_D6*
CCD Application	Active
Clock Pin Placement	Default

Table 3.5: Block 1, Block 2 and Block 3 default parameters.

All the experiment sets performed after the initial set have been done by modifying one of the given parameters unless it is specified on the experiment set definition.

3.5.2. Initial Experiment Sets

The initial approach from which the project was to work upon the configuration options of the tool.

Compared to modifications in a block done by an engineer, modifications done in the application options of a design tool do not affect as heavily the physical design of a block.

To limit the scope of the project, the application options were intended to focus only on the Clock Tree Synthesis of the block.

Given modifications on the application options of the design tool, it is desired to optimize the design based around a given set of blocks with different physical characteristics.

The initial application options experiment sets that were considered were based around the following considerations:

- *Concurrent Clock and Data Optimization*
- Layer Optimization
- Integrated Clock Gater Optimization
- Congestion Effort
- Clock Miscellaneous Configurations
- *Local skew* Optimization
- Other Configuration Options

3.5.2.1. Configuration options explanations and expected results

The explanation for the configuration options given before is the following one:

- *Concurrent Clock and Data Optimization*: *Concurrent Clock and Data Optimization* (*CCD*) is a configuration option introduced by *Synsopsys* to help meeting timing constraints on high frequency circuits.

  The main application options that were covered in this experiment sets revolved around enabling *Concurrent Clock and Data Optimization* were:

  a. *CCD* enabling during placement, clock routing and general routing.
  b. Boundary timing enabling: *Concurrent Clock and Data Optimization* can either be applied to all the flip-flops in the design or only to non-boundary flip flops. Omitting boundary flip-flops is useful when signals can be introduced from outside, pin capacitance may not be stable or controlled and thus omitting them from being applied to *CCD* can be benefitial.
  c. Clock aware placement: Enables placement of Integrated Clock Gaters (*ICG*) and their *fanout* cells considering timing criticality.

- Layer Optimization: Layer optimization considers layer resistance and capacitance, as well as possible congestion problems to manage how routing is distributed through different layers.

a. Placement and clock routing layer optimization: If critical nets are detected, those are assigned to upper metal layers to improve timing.
b. Layer overlapping: Metal layers are set in pairs, this option enables mixing the metal layer pairs to improve the *QoR* obtained.

- Integrated Clock Gaters Optimization: Integrated Clock Gaters are clock cells designed to switch off certain parts of the clock tree to avoid unnecessary power consumption in certain scenarios.

a. Clock aware placement and trial clock tree: Enables an early clock tree during placement used for *ICG* optimization flow.
b. *ICG* auto bound: Creates group bounds for *ICG*s automatically for placement.
c. *ICG* placement optimization: Enables *ICG* optimization with the specified options.
d. *ICG* splitting: Enables *ICG* splitting to meet timing constraints unless specified otherwise.

- Congestion Effort: Congestion is specified as the wiring resources used on a metal layer compared to the total resources available. It must also be considered local congestion where it may have over-usage of wiring resources. A high congestion effort will typically trade runtime against overall layer congestion.

a. Placement congestion effort.
b. Clock routing congestion effort.

- Clock Miscellaneous Configurations

This set of configuration options considers, hold meeting effort, placement effort and power recovery.

a. Hold fixing effort.
b. Clock coarse placement effort.
c. Clock Power Recovery: Enables power or area recovery in clock and general routing.

- Other Configuration Options: This last set considers several configuration options regarding clock routing, timing, power and area optimization.

a. Timing effort optimization: Optimization effort on timing enclosure in placement and clock routing.
b. Area effort optimization: Area recovery effort in placement and clock routing.
c. Placement path optimization.

d. Power mode optimization: Enables leakage power, dynamic power or total power in pre-routing.
e. Power effort optimization: Controls the effort when power mode optimization is enabled.

3.5.3. Slew Analysis

As explained in the Project Scope Changes regarding structural modifications, one of the variations considered is the *slew* constraint.

Slew indicates which the maximum signal transition time at the input of an inverter is. Due to the clock wiring tracks *RC* Elmore delay, the admitted *slew* correlates directly with the buffer distance separation.

The slew selection is done in order to minimize process variability and clock jitter. Process variability is inherent in manufacturing processes and must be accounter for possible variations on the characteristics of transistors in a circuit.

Jitter is the variation on the rising and falling edges of a clock from their ideal values. One of the causes of jitter is signal noise, which tends to be more relevant as power supply voltage is reduced.

By using a more restrictive slew constraint, the effects of process variability and jitter can be reduced.

In terms of jitter, by placing repeaters closer to each other, the possible mismatches in signal transition and thus jitter are reduced.

Slew selection is done considering On-chip variation (OCV). On-chip variation is defined as variations in the design due to manufacturing processes such as etching or oxide thickness. In modern design and manufacturing techniques this problems will happen and must be taken into consideration.

Process variability refers to the range on which for example the transistor width or channel length that can vary from transistor to transistor. Voltage variations can be due to multiple power supplies in a block or from other causes like IR-drop. IR-drops are related to the current driven and the resistance of the track which will cause a voltage drop. IR-drop effectively modify the voltage seen at any part of the circuit and must be taken into consideration. Finally temperature affects performance on a circuit. It is likely to assume that different parts of a block will have different power consumption. Higher power consumption will result on higher chip dissipation and thus higher temperatures. Higher temperature in a clock cell will usually result on it being slower; on the other hand, a lower cell temperature will result on it being faster.

By selecting a correct *slew* constraint it is ensured that the *repeater*s are placed close enough to minimize the effects of process variability and clock jitter.

- The *slew* constraint is smaller than the optimal one: The *repeater*s will be placed too close resulting on timing metric degradation and power increases due to the increased number of *repeater*s used.
- The *slew* constraint is bigger than the optimal one: Targets across the corners will not be met.

3.5.3.1. Experiment Set Definition

The *slew* analysis has been performed in the initial test block and the analysis blocks. The target is the optimization of the *QoR* metrics. Several *slew* constraints have been selected for each block.

The following table summarizes which are the *slew* constraints used on each block. The *slew* constraints used in Block 3 have been derived from the results obtained on the first two blocks.

Slew constraint (ps)	100	80	70	60	40
Block 1	✓	✓	x	✓	✓
Block 2	✓	✓	x	✓	✓
Block 3	✓	✓	✓	✓	✓

Table 3.6: Experiment set definition table for the *slew* constraint experiment set at Block 1, Block 2 and Block 3.

The *slew* analysis will be done block by block remarking the most noticeable aspects of the experiment sets performed on each block. The complete results will be presented on the annexes.

3.5.4. Fanout Analysis

One of the main points that were considered as modifications to test was the maximum clock *fanout* constraint.

On typical zero-*skew* balance clock tree structures, the increase of the clock *fanout* limit usually results on a clock tree with less intermediate *repeater*s and higher grouping of clock cells at the end-point cells that connect to the flip-flops of the design.

This reduction on *repeater*s can provide the following benefits:

- Reduction at the worst case *insertion delay*.
- *Skew* improvement: By simplifying the clock structure there are less intermediate *repeater*s. This increases the common path between flip-flops that have a common datapath which in terms results on a reduction of the *skew*.
- Reduction of the *Worst Negative Slack*: By reducing the intermediate clock cells in the clock structure the disparity between different clockpaths which results on a overall better WNS.
- Clock power reduction: The reduction on the number of *repeater*s will result on reductions of the dynamic and leakage clock power.

It should be noted that ideally, heavily increasing the *fanout* constraint will require an increase on the *repeater* cell size.

As it has been explained, the usage of *Concurrent Clock and Data Optimization* set of options breaks the expected balancing around the modification of the *fanout* limit.

It has to be considered as well the physical layout of the used block. Flop and memory placement can affect heavily how the modifications at the clock *fanout* will affect the overall design.

3.5.4.1. Experiment Set Definition

The base *fanout* configuration selected for all the blocks was 32. The initial analysis has been done only modifying the *fanout* without making further modifications to other parameters.

The following table summarizes which *fanout* constraints were used on each block.

Fanout constraint	32	64	128	256	512
Block 1	✓	✓	✓	✓	✓
Block 2	✓	✓	✓	✓	✓
Block 3	✓	✓	✓	✓	x

Table 3.7: Experiment set definition table for the *fanout* constraint experiment set at Block 1, Block 2 and Block 3.

In all cases, *fanout* selection should be done considering the physical layout and should be expected to be different from block to block.

Similar to the *slew* analysis the full data will be on the annex while only the most relevant results on each block will be presented.

The analysis will be done based on the *Quality of Results* extracted and information on the clock tree structure.

Due to the results obtained on Block 1 and Block2 block, the *fanout* analysis at 512 was not done at Block 3

3.5.5. Clock Cell Analysis

After the experiments regarding *slew* and *fanout* were done the next step is analysing the selection of invereters in the *repeater* selection.

When considering the selection of *repeater* cells for the clock tree only inverters have been considered. The use of inverters is preferred over buffers due to the inherent reduced delay.

Typically, inverters will have half the parasitic capacitance and thus transition delay of a buffer using the same technology. However, when using a inverter it must be considered the phase when the signal arrives at the desired point. On this case, the addition of an additional inverter must be considered to invert the phase. This consideration is usually not necessary when using buffers as the input buffer phase is the same as the output buffer phase, whereas on inverters, the input and output phase are the opposite.

When selecting clock cells several considerations must be taken. The usage of big clock cells will have the following consequences in the clock tree:

- Increased cell parasitic capacitance and cell delay.
- Increased cell dynamic power.
- Increased cell leakage power.
- Increased driving power.

On the other hand, it should be expected that the use of small clock cells will result in:

- Reduced cell parasitic capacitance and cell delay.
- Reduced cell dynamic power.
- Reduced cell leakage power.
- Reduced driving power.

It is understood as driving power, the capability of a clock cell to propagate a single change to its output *fanout* correctly.

An insufficient driving power will result on not being able to switch the clock signal at each of the clock cells driven by the input driver. A higher *fanout* at the output of a *repeater* will require of higher driving power, and a low *fanout* will require low driving power.

Asides from these variations when different size cells are used, other considerations must be done regarding clock tree building.

As explained before the flow used has activated by default the *Concurrent Clock and Data Optimization* (*CCD*) configuration option.

This configuration option builds the clock tree considering the datapath. With this option active, it is sought to minimize the *Worst Negative Slack* of the design. This differs from more typical zero-*skew* clock building technique where the key aspect is balancing each tree branch to have the same delay from the clock pin to each of the flip-flops at the end of the clock to reduce the *skew*.

It is to be expected then, that the usage of *CCD* modifies the expected behaviour obtained when modifying the *repeater*s used, compared to a zero-*skew* balancing technique.

In a zero-*skew* balancing technique, the modification of the clock cells in a tree branch will result on applying the same modification in all the tree branches to keep the tree balance.

Once *Concurrent Clock and Data Optimization* is applied, given the *Worst Negative Slack* minimization strategy, several additional conditions must be considered.

The change of the clock tree wirelength, number of clock cells and even variation of parts of the structure may be needed to keep the clock balance that yields the best results in terms of *Worst Negative Slack*.

To build the clock tree three inverters are used. All the inverters are used on clock tree building while the two smallest inverters are used on the balancing step.

Several experiment sets have been selected and performed regarding *repeater* usage. The first experiment done involves the usage of bigger clock cells while the second experiment is focused on the usage of smaller clock cells.

As seen before, the usage of bigger or smaller clock cells is correlated with the driving power needed and thus the output *fanout* a *repeater*. Following this consideration, when bigger clock cells have been used, they have been paired with larger *fanout* constraints. On the other hand, when smaller clock cells have been used, the *fanout* constraint has been more controlled with lower *fanout* constraints.

3.5.5.1. Experiment Set Definition

The sets of clock cells used are the following ones.

- Reference *repeater* set:
  a. Clock Tree Building: *INV_D16, INV_D12, INV_D6*
  b. Clock Tree Balancing: *INV_D12, INV_D6*
- Bigger *repeater* set:
  a. Clock Tree Building: *INV_D24, INV_D16, INV_D8*
  b. Clock Tree Balancing: *INV_D16, INV_D8*
- Smaller *repeater* set:
  a. Clock Tree Building: *INV_D14, INV_D10, INV_D4*
  b. Clock Tree Balancing: *INV_D10, INV_D4*

The experiment sets selected for each block are summarized on the following tables:

Block 1	*Fanout* = 32	*Fanout* = 64	*Fanout* = 128	*Fanout* = 256	*Fanout* = 512
Bigger *repeaters*	✓	X	X	X	✓
Smaller *repeaters*	✓	✓	X	X	X

Table 3.8: Experiment set definition table for the clock cell selection with *fanout* variation constellation at Block 1.

Block 2	*Fanout* = 32	*Fanout* = 64	*Fanout* = 128	*Fanout* = 256	*Fanout* = 512
Bigger *repeaters*	✓	X	X	✓	X
Smaller *repeaters*	✓	✓	X	X	X

Table 3.9: Experiment set definition table for the clock cell selection with *fanout* variation constellation at Block 2.

Block 3	*Fanout* = 32	*Fanout* = 64	*Fanout* = 128	*Fanout* = 256	*Fanout* = 512
Bigger *repeaters*	✓	**X**	✓	✓	**X**
Smaller *repeaters*	✓	✓	**X**	**X**	**X**

Table 3.10: Experiment set definition table for the clock cell selection with *fanout* variation constellation at Block 3.

The first two blocks were done following the same constellations regarding *fanout* and clock cell. For Block 3, the results and analysis on the first two blocks was done and it was modified from the conclusions extracted.

Finally and due to problems in the selection of clock cells, a run in Block 3 was done using several *repeaters*.

This last run used both the reference and the smaller *repeater* sets and it will be covered due to the results obtained.

3.5.6. Concurrent Clock and Data Optimization analysis

In the development of the blocks, there is a set of options provided by *Synopsys* called *Concurrent Clock and Data Optimization*. This set of options is designed in order to achieve maximum frequency on high-performance designs.

Concurrent Clock and Data Optimization provides *skew* optimization and datapath optimization to maximize the maximum frequency, and thus minimize the *Worst Negative Slack*, of a design.

This set of options correlates the datapath and the clockpath in order to obtain the best timing metrics. This however presents several both in the clock building strategy and the expected behaviour in terms of metrics obtained when some changes are applied in a design.

Typical clock building strategies use a zero-*skew* algorithm to generate the clock tree. This type of structure tends to be simpler in times of clock building and have much more predictable results in terms of metrics.

All the blocks analysed are configured to use *Concurrent Clock and Data Optimization* by default. The experiment sets were configured to analyse the results obtained when this application option set was deactivated under certain conditions.

The clock building step done at the company base flow can either use a The clock building step done at the company base flow can either use a *Synopsys* compact command or a custom set of commands to build and optimize the clock tree.

In terms of configuration options there is also the possibility of using *local skew* optimization or not.

Skew optimization can be done either globally or locally. Global *skew* is defined as the timing difference between the earliest reaching flip-flop and the latest reaching flip-flop on a block.

On the other hand, *local skew* optimization is the timing difference between the earliest arrival and the latest arrival between flip-flops in the same clock path.

Global *skew* optimization can sometimes be inaccurate as it may be the maximum timing between two flip-flops that do not share clock path and thus the timing difference has no meaning in terms of clock arrival times.

3.5.6.1. Experiment Set definition

The experiment sets on this case have been designed to account for possible variations when *Concurrent Clock and Data Optimization* configuration options are not applied.

The sets chosen are the following ones:

- Application of *Concurrent Clock and Data Optimization*.
- *Concurrent Clock and Data Optimization* not applied with *Synopsys* clock routing command and *local skew* optimization.
- *Concurrent Clock and Data Optimization* not applied using a custom clock tree building command set and *local skew* optimization.
- *Concurrent Clock and Data Optimization* not applied with *Synopsys* clock routing command without *local skew* optimization.

This experiment set has been tested on Block 1 and Block 2. The analysis on Block 3 has not been performed due to the results obtained on the first two blocks.

3.5.7. Clock Placement Analysis

In all the experiment sets that have been defined in this section, the placement of the clock pin is the reference one given.

From the results obtained it has been seen that the clock building algorithm is done through an iterative propagation process from the input clock pin.

In all the blocks that have been defined, the application of *Concurrent Clock and Data Optimization* has been done.

From the analysis done regarding *Concurrent Clock and Data Optimization* it has been seen that it is mandatory for high frequency circuits and gains even more importance on circuits with a constrained block in terms of memory instances and placement.

When using this application option set, the clock algorithm will be focused on minimizing the *Worst Negative Slack* and thus it does not focus on traditional clock balance metrics such as *skew* or *local skew* minimization.

The experiment set that will be performed will focus on the movement of the clock input pin to the centre of the block entering by one of the upper metal layers used by clock on the different blocks.

On this experiment set it will be checked if it is possible to optimize metrics such as *insertion delay* and *skew* in the design while using *Concurrent Clock and Data Optimization* and thus optimizing the *Worst Negative Slack*.

On the reference cases, the clock input pin is placed in a side of the block, thus by centring the clock pin it should be ideally reduced the maximum clock wirelength from the clock pin to the furthest flip-flop of the design.

This speculation is done without considering memory placement or the behaviour of *Concurrent Clock and Data Optimization* algorithm and thus it will be tested which are the results obtained and how they differ from the expected results presented on this section.

3.5.7.1. Experiment Set Definition

To simplify this analysis it has only been done on the blocks that yielded the best results. Taking into account all the analysis but the first one, focused on configuration options, that was discarded due to reasons previously presented, the blocks upon this last experiment will be performed are the following one:

- Block 1:
  c. *Fanout* constraint: 32
  a. *Slew* constraint: 80ps
  b. *Repeater* cells: *INV_D16, INV_D12* and *INV_D6*
  c. *Concurrent Clock and Data Optimization* options: Active

- Block 2: Optimal timing metrics
  a. *Fanout* constraint: 32
  b. *Slew* constraint: 80ps
  c. *Repeater* cells: *INV_D16, INV_D12* and *INV_D6*
  d. *Concurrent Clock and Data Optimization* options: Active

- Block 2: Optimal power metrics
  a. *Fanout* constraint: 64
  b. *Slew* constraint: 80ps
  c. *Repeater* cells: *INV_D16, INV_D12* and *INV_D6*
  d. *Concurrent Clock and Data Optimization* options: Active

- Block 2: Reference block
  a. *Fanout* constraint: 32
  b. *Slew* constraint: 80ps
  c. *Repeater* cells: *INV_D16, INV_D12* and *INV_D6*
  d. *Concurrent Clock and Data Optimization* options: Active

- Block 3: Optimal power metrics
  a. *Fanout* constraint: 128
  b. *Slew* constraint: 80ps
  c. *Repeater* cells: *INV_D16*, *INV_D12* and *INV_D6*
  d. *Concurrent Clock and Data Optimization* options: Active

- Block 3: Optimal timing metrics
  a. *Fanout* constraint: 32
  b. *Slew* constraint 80ps
  c. *Repeater* cells: *INV_D16, INV_D12* and *INV_D6*
  d. *Concurrent Clock and Data Optimization* options: Active

- Block 3: *Slew* modified Block
  a. *Fanout* constraint: 128
  b. *Slew* constraint: 100ps
  c. *Repeater* cells: *INV_D16, INV_D12* and *INV_D6*
  d. *Concurrent Clock and Data Optimization* options: Active

4. Results

This section will cover the analysis done on each of the experiment sets defined in the previous section. The data obtained on all the experiment sets performed will be added on the appendices and only some of the most relevant data will be presented here.

4.1. Initial Experiment Sets

The initial experiment sets were analysed in Block 1, Block 2 and the Test Block. From these blocks, it will only be considered the results obtained on Block 1 and Block 2.

The results obtained on the test block will be added to the annex but will not be discussed here because:

- Block is overly simple compared to the other blocks as it can be seen on the block definition section.
- Block was initially assessed only as a test block, thus all the experiments are run first on the test block to see if they are executed correctly, without considering the obtained results as valid.

Although the Test Block was defined only to check for possible errors or problems for the experiment set and to tests the different scripts generated.

On this experiment set, it was used to reduce the experiment sets performed at Block 1 and Block 2 to reduce the runtime and resource usage whenever possible.

Thus, from the initial experiment sets, the ones that were analysed in Block 1 and Block 2 were:

- Layer Optimization.
- Integrated Clock Gaters Optimization.
- Congestion Effort.
- Local skew Optimization.

Given the simplicity of the Test Block where the experiment sets were performed, several conditions were imposed in order to select the final experiment sets to be performed on Block 1 and Block 2.

First of all, it was considered mandatory that all blocks met the timing enclosure, having a zero or positive *Worst Negative Slack*. Asides from this condition, it was considered positive to have improvements in either:

- Skew and insertion delay.
- Clock and general power.
- Clock area and number of instances.
- Utilization.

4.1.1. Block 1 analysis

From the selected experiment sets selected before, all configuration sets but the *local skew* configuration sets yielded much worse results in terms of *Worst Negative Slack* with different results in *Total Negative Slack*. The difference in *Total Negative Slack* depends on the number of violating paths.

In general terms, *skew* improves in all experiment sets while *latency* varies depending on the experiment sets. The optimization of *ICG*s and the modifications on the *local skew* configuration help improving the *insertion delay* while the relaxation of the constraints regarding congestion and the usage of layer optimization yielded worse results.

In terms of *repeater*s used, all experiment sets use more clock cells, having the highest peak in the *local skew* experiment set. However, with the relaxation of the congestion configuration options, fewer cells are used as it is expected.

Regarding power, all runs but congestion effort yield a higher clock power as it is expected from the increase on the additional *repeater*s used on the clock network.

In terms of total power, all blocks suffer from the increased power derived from the clock network, varying from increases ranging between 1.5% and 5% depending on the block.

4.1.2. Block 2 analysis

Compared to the other blocks, the WNS and TNS are overall worse in all blocks by a huge margin. *Insertion delay* in all cases also increases while *skew* depends on the block.

Overall, regarding the usage of *repeater* cells, relaxing the congestion constraints of the tool, yields mild decreases on cell instances and thus clock power. The other blocks, following the trends observed on Block 1, suffer from increases on the clock power.

In terms of general power, the total power decreases only while relaxing the congestion effort while in all other cases increase.

4.2. Change on the Original Approach

In the initial experiments done it was not possible to attain improvements in power or timing amongst other metrics without incurring in degradation in other metrics.

However the main problems regarding this set of experiments does not account by the *Quality of Results* obtained but for how the methodology was performed.

The initial experiment sets were designed to focus on the configuration options provided by the tool.

Several sets were made where several application options of the tool were grouped depending on the function they performed as it has been explained on chapter 3.

Several results have been obtained although they have not been conclusive, giving different results in both Block 1 and Block 2 where they were tested.

One of the main conclusions obtained in the development of this block is that even when applying a group of application options that are correlated with each other, for example layer optimization, it is impossible to discern which of the options modified accounts for the variations obtained in the metrics.

To perform a full test regarding application options, it would have been necessary to generate a test for each of the application options on each of the blocks which would result on a number of experiment sets and analysis out of the scope of this master thesis.

With this result in mind, the change of approach has been focused around more controlled modifications, this case regarding parameters of the clock structure with a more limited approach.

4.3. Slew Analysis

The analysis done on each block is specified in the tables in the experiment set definition section. In this section it will be analysed the results obtained, which are presented on the appendices.

The reference conditions upon which the analysis has been performed on this block are the following ones:

- Reference clock *fanout* constraint: 32
- Reference *slew* constraint: 80 ps
- Reference set of *repeater* cells: *INV_D16, INV_D12* and *INV_D6.*
- Application of *Concurrent Clock and Data Optimization*: Active
- Position of clock pin: Reference position

4.3.1. Block 1 analysis

As seen on the block characteristics, this block is memory dominated. There is a total of 417 memories distributed across all block. The flops are placed uniformly in the block and in-between memories.

The *slew* analysis has been performed in 4 values; 100ps, 80ps, 60ps and 40ps. The 80ps block is the reference block.

In all the blocks analysed, the *Worst Negative Slack* has worse results, up to 47% at *slew* constraint of 40ps, while the *Total Negative Slack* worsens by 58% on that same run.

Regarding *insertion delay*, only the run at a *slew* reference of 60ps, gives better results, being 1.207 ns compared to the 1.37 ns obtained on the reference run.

In terms of *repeater* increase and power the results obtained are the following ones:

As the *slew* constraint is reduced, the number of clock *repeater*s increases on an exponential rate, following a similar trend to the clock power. At 460 ps, the number of clock *repeater*s used increases by 7% while at 40 ps this percentage increases up to 30% compared to the reference case. The clock power increases by 3.3% at 60 ps, increasing up to 6.1% at 40ps.

Considering the clock wirelength, in all the cases where *slew* is analysed, the clock wirelenfth increases, up to a 12% compared to the reference case, accounting for a part of the dynamic power increase obtained in all the *slew* cases analysed.

Due to the increases on clock power, the total dynamic power suffers increases of up to 4% at a *slew* constraint of 40 ps.

The number of *repeater*s, and power used for each *slew* constraint is:

Slew constraint	100ps	80ps	60ps	40ps
Number of *repeater*s	30,718	28,969	31,044	3,7598
Clock Wirelength (µm)	1,815,786.69	1,682,771.29	1,815,399.50	1,886,413.97
Dynamic Power (µW)	746,510.6	722,639.7	746,462.8	766,330.6
Leakage Power (µW)	1,400.74	1,329.53	1,392.47	1,617.62

Table 4.1: Clock power and number of *repeater*s used and clock wirelength for the *slew* variations considered at Block 1 for a logic activity factor of 10% and clock activity factor of 200%.

As it can be seen, the number of *repeater*s increases from 31044 to 37598 *repeater*s when reducing the *slew* constraint from 60ps to 40ps, while when decreasing the *slew* constraint from 80ps to 60ps, the number of *repeater*s only increases from 28969 to 31044.

This trend is followed in the leakage power, which only depends on the clock cells increasing from 1329.53µW to 1392.47µW when lowering *slew* from 80ps to 60ps, while it increases from 1392.47µW to 1617.62µW when lowering the *slew* from 60ps to 40ps.

When general metrics are regarded, the total dynamic power is driven up by the clock power increase. On average, the dynamic power increase correlates correctly with the total power increase.

From the reference block and all the *slew* analysis it can be seen that the clock dynamic power represents 49.5% of the total power. When a *slew* constraint of 40ps is applied, the clock power contribution to the total dynamic power rises to 50.8% due to the increased number of *repeater*s and clock wirelength compared to the 49.1% of the reference block.

Regarding the clock tree structure the smallest inverters used in the clock tree structure suffer the highest absolute increase. The total results regarding *repeater* distribution by size are the following ones:

Slew constraint	100ps	80ps	60ps	40ps
INV_D16	1,395	2,172	1,327	1,053
INV_D12	5,379	3,630	4,921	6,181
INV_D6	23,944	23,167	24,796	30,364

Table 4.2: *Repeater* breakdown distribution at Block 1 for the different *slew* constraints analysed.

Slew constraint	100 ps	80ps	60ps	40ps
Average *repeater fanout*	14	15	14	12
Average *ICG fanout*	8	5	5	1
Cells *fanout* =< 16	21,446	20,264	21,730	28,318
Cells *fanout* => 32	10,790	11,078	10,768	10,598

Table 4.3: *Repeater fanout* breakdown distribution and average clock cells *fanout* at Block 1 for the different *slew* constraints analysed.

From Table 4.2 it can be seen that the average *ICG fanout* drops to 1 when a *slew* constraint of 40ps is applied, possibly indicating errors in the clock structure, thus the results obtained should be treated carefully.

Overall, the *repeaters* *INV_D16* are reduced from 2172 instances to 1053 at a *slew* constraint of 40 ps compared to the reference value of 80 ps. On the other hand, the intermediate and smallest *repeaters* suffer from the biggest increases, going up from 3630 to 6181 instances for *INV_D12* and increasing from 23167 to 30364 instances for *INV_D6.*

This increase on the number of *repeaters* can be seen in the *repeater* breakdown, where the average *repeater fanout* tends to decrease while the low *fanout* cells increase from 20264 instances to 28318.

4.3.2. Block 2 analysis

As done in the previous block, *slew* analysis was done at 100ps, 80ps, 60ps and 40ps. Similarly, the 80ps block is the reference run with the base values.

Regarding the *QoR* metrics obtained on this block, the *Worst Negative Slack* has worse results in all the blocks analysed, going up from -0.06 ns to -0.134 ns at a *slew* constraint of 100 ps, and having worse results in all the other cases analysed. The *Total Negative Slack* shows a similar trend. It must be noticed however the reduction on the number of violating paths at a 40 ps *slew* constraint, going down from 163 to 98.

The *insertion delay* increases as the *slew* constraint is reduced, increasing from 0.762 ns to 0.799 ns at 40 ps.

In all cases the number of *repeaters* added suffers from a similar increase as the one expected at the Block 1 *slew* analysis.

When lowering the *slew* from 80ps to 60ps the number of *repeaters* increases from 18036 to 20560, while when lowering the *slew* from 60ps to 40ps the number of *repeaters* increases from 20560 to 32995.

Considering the clock power consumption, the dynamic power suffers an increase from 429350.79µW to 460323.18µW when lowering the *slew* from 80ps to 40ps, while the leakage power increases from 962.96µW to 1533.17µW.

Once again, the leakage power follows similar increases to the ones experienced in the number of *repeater*s while the dynamic power is much more relaxed due to the increase from the number of *repeater*s only being a part of the total clock dynamic power.

From these results it can be seen that halving the *slew* from 80ps to 40ps causes an increase on the number of *repeater*s of 79.05%. Assuming an exponential trend, further lowering the *slew* constraint would cause a much greater increase on the number of *repeater*s coupled with a degradation of the power and timing metrics even more pronounced.

Slew constraint	100ps	80ps	60ps	40ps
Number of *repeater*s	18,733	18,036	20,560	32,295
Dynamic Power (µW)	436,811.42	429,350.79	433,174.58	460,323.18
Leakage Power (µW)	992.23	962.96	1,091.38	1,533.17

Table 4.4: Clock power and number of *repeater*s used for the *slew* variations considered at Block 2 for an activity factor of 10%.

Data regarding clock structure was also obtained. The most relevant information when *slew* is considered is the number of instances and how is distributed, as well as the average *repeater* and *ICG fanout*.

Slew constraint	100ps	80ps	60ps	40ps
INV_D16	723	691	993	1295
INV_D12	2,449	2,270	3,101	4,624
INV_D6	15,561	15,075	16,466	26,376

Table 4.5: *Repeater* breakdown distribution at Block 2 for the different *slew* constraints analysed.

Slew constraint	100ps	80ps	60ps	40ps
Average *repeater fanout*	14	14	13	9
Average *ICG fanout*	9	9	9	7
Cells *fanout* =< 16	18,182	17,518	19,968	31,830
Cells *fanout* => 32	8,058	8,156	7,836	6,975

Table 4.6: *Repeater fanout* breakdown distribution and average clock cells *fanout* at Block 2 for the different *slew* constraints analysed.

The average *fanout* follows a similar trend where it decreases from 14 at 80ps to 9 at 40ps while the *ICG* decreases from 9 to 7. On this case, compared to Block 1, it is not observed an extreme reduction on the average *ICG* at 40 ps.

The inverter *INV_D16* almost doubles its instances when reducing the *slew* constraint from 80 ps to 40 ps, increasing from 691 instances to 1295 instances. This trend is followed in all the other inverters, increasing from 2270 to 4624 for *INV_D12* and from 15075 to 26376 for *INV_D6.*

The biggest inverter suffers almost doubles from 80ps to 40ps, however the most relevant increases are on the *INV_D6_N* cell increasing from 15075 at 80ps instances to 26376 instances at 40ps.

Considering the *repeater*s connected to the flip-flops, the number of high *fanout* nets decreases from 8156 to 6975 while the number of *repeater*s with a *fanout* equal or lower than 16.

4.3.3. Block 3 analysis

The *slew* analysis for Block 3 was done after the *slew* data and analysis on Block 1 and 2 was done.

The *slew* constraints selected on this case has been expanded, having a analysis at 100 ps, 80ps, 70ps, 60ps and 40 ps.

To check which the optimal *slew* for this case is several parameters have been analysed:

- User-defined Quality of Result metrics.
- Number of violating paths.
- Clock structure physical information.

The *Quality of Results* will be the main tool to assess which *slew* constraint yields the best results while the number of violating paths and the clock structure information will be used to give a better insight on the results obtained.

The number of violating paths can be used to assess in this case how many clock cells suffer from *slew* violations depending on the constraint applied in each case. It should be expected less *slew* violations as the *slew* constraint becomes tighter.

The *Worst Negative Slack* and *Total Negative Slack* can be omitted from the analysis as timing closure is not critical for Block 3. The worst *WNS* is obtained at a *slew* constraint of 100ps with a value of -0.007 ns.

The best results regarding *slew* and *insertion delay* are obtained at 100ps and 40ps. On the first case the result obtained is due to the reduction on the number of *repeater* instances used, which decreases by 20%. At 40ps, the reduction on the *skew* and *latency* is probably due to a better clock structure as the number of instances increases.

Regarding the usage of inserters, similar results to the ones obtained on Block 1 and Block 2 are observed. The number of instances increases as the *slew* constraint decreases. At 40ps the number of instances used increases up to 50%.

The clock power follows a similar trend where it increases as the number of clock *repeater*s grows. At 70 ps, despite the increase on the number of instances used, the reduction on the clock wirelength, results on a reduction of 0.2% on the clock power.

Regarding general power, the dynamic power is determined by the dynamic clock power variation while the leakage power tends to increase as the *slew* constraint is reduced.

Regarding the clock structure on Block 3 the results obtained are the following:

Slew constraint	100ps	80ps	70ps	60ps	40ps
INV_D16	268	433	481	449	626
INV_D12	1,388	1,623	1,550	1,674	2,704
INV_D6	6,840	8,513	9,462	10,082	12,074

Table 4.7: *Repeater* breakdown distribution at Block 3 for the different *slew* constraints analysed for a *fanout* constraint of 32.

Slew constraint	100ps	80ps	70ps	60ps	40ps
Average *repeater fanout*	19	16	15	15	12
Average *ICG fanout*	13	13	12	12	12
Cells *fanout* < 16	12,954	15,582	16,870	17,399	20,273
Cells *fanout* > 32	4,980	4,885	4,862	4,387	4,327

Table 4.8: *Repeater fanout* breakdown distribution and average clock cells *fanout* at Block 3 for the different *slew* constraints analysed for a *fanout* constraint of 32.

In terms of *repeater* selection, the number of *repeater*s remains approximately constant up to 40 ps where the number of instances of each instances suffers bigger increases.

- *INV_D16*: from 433 instances to 626.
- *INV_D12*: from 1623 instances to 2704.
- *INV*_D6: from 8513 instances to 12074.

The average *repeater fanout* tends to decrease as the number of *repeater*s used increase, similarly to the results obtained in the previous blocks while the average *ICG fanout* remains approximately constant.

Regarding the *fanout* breakdown, the number of clock cells with a *fanout* lower than 16 increases, as shown in the *repeater* breakdown as the *slew* constraint decreases. Considering that the number of flops is fixed, this increase results on a reduction on the number of higher *fanout* cells.

4.4. Fanout Analysis

As it was done for the *slew* analysis, the *fanout* experiment sets are the ones defined at the tables in the experiment set definition and have been performed on Block 1, Block 2 and Block 3.

The reference parameters used on this experiment set are the following:

- Reference clock *fanout* constraint: 32
- Reference *slew* constraint: 80 ps
- Reference set of *repeater* cells: *INV_D16, INV_D12* and *INV_D6.*
- Application of *Concurrent Clock and Data Optimization*: Active
- Position of clock pin: Reference position

4.4.1. Block 1 Analysis

Considering the *Quality of Results* analysis it can be seen that:

The number of *DRC* violations at a *fanout* constraint of 64 increases by 50% approximately and thus the results obtained on this run could contain errors and should be considered for any analysis done on it.

The reference run with the smallest *fanout* has the best results regarding *Worst Negative Slack*, yielding increases of up to 63% in the worst case.

The *skew* and *latency* tend to improve as the *fanout* constraint is increased. The improvements of *skew*, up to 17% for the highest *fanout* constraint, is due to the increase on the common path between flip-flops while the improvements of *latency*, decreasing from 1.37 ns to 1.213 at a *fanout* constraint of 512, is due to the reduction on the number of *repeater*s used.

Considering then number of instances used, it can be seen that the best result in terms of instance reduction is observed at a *fanout* constraint of 128 with improvements of 26%, compared to the 20% improvement at a *fanout* constraint of 512.

The reduction on the clock area follows a similar rend to the reduction on the numbers of *repeater*s used.

Despite the reductions on the number of clock cells used, the dynamic power is only reduced when a *fanout* constraint of 128 is used. When analysing the clock wirelength it can be seen that for all the *fanout* cases analysed, there are increases on the wirelength at a *fanout* constraint of 64, 256 and 512, compared to the reference case.

In terms of general power consumption, the total dynamic power increases in all blocks but for 128 constraint, driven by the power increases of the clock tree. On the other hand, the total leakage power increases up to 18% in the worst cases.

The results obtained regarding clock structure are summarized in the following tables:

Fanout constraint	32	64	128	256	512
INV_D16	2,172	1,206	1,157	1,203	1,254
INV_D12	3,630	4,671	3,774	4,013	3,862
INV_D6	23,167	24,675	16,402	19,131	17,811

Table 4.9: *Repeater* breakdown distribution at Block 1 for the different *fanout* constraints analysed.

Fanout constraint	32	64	128	256	512
Average *repeater fanout*	15	14	19	17	18
Average *ICG fanout*	5	7	7	7	7
Cells *fanout* =< 16	20,264	21,222	17,630	20,619	19,138
Cells *fanout* => 32	11,078	10,755	5,944	5,920	5,933
Cells *fanout* => 64	0	0	2,632	2,628	2,576
Cells *fanout* => 128	0	0	42	45	36
Cells *fanout* => 256	0	0	0	0	0

Table 4.10: *Repeater fanout* breakdown distribution and average clock cells *fanout* at Block 1 for the different *fanout* constraints analysed.

The main results to take into consideration are:

- From 128 and upwards, the *fanout* limit is not achieved.
- As *fanout* increases, the biggest inverters and the smallest ones are used less while the middle sized inverters are used more.
- The average *repeater fanout* increases up to 128, upon which it starts to decrease again.

As the *repeater fanout* increases, the average *repeater fanout* tends to increase, excluding the *fanout* constraint of 64 case that could contain errors, up to 128 where it starts to decrease again.
Similarly, the average *ICG fanout* increases from 5 to 7 as the maximum *fanout* constraint increases.
From the results in Table 4.10 it can be seen that the *fanout* constraints of 256 and 512 are not achieved as no clock cells achieve those *fanout* constraints. Similarly, only 42 cells achieve a *fanout* constraint of 128 when the constraint is set at said value. From these results the use of *fanout* constraints larger than 64 should be avoided.
As said before, it is possible that the run using a *fanout* constraint of 64 had errors. This can be checked as for this run, no cells using a *fanout* constraint of 64 use this *fanout*.
In terms of clock optimization, the use of *fanout* constraints that cannot be achieved due to block layout or other constraints should be avoided as it may yield a degraded clock structure.

4.4.2. Block 2 Analysis

Regarding metric analysis the main points regarding the *Quality of Results* are the following ones.

The reference run with a *fanout* of 32 and the block with a *fanout* of 64 constraint yields the best results in terms of *Worst Negative Slack*, -0.06 ns and -0.07 ns respectively, while on the other cases degrade up to -0.165 ns with a *fanout* constraint of 512.

The *skew* obtained in all blocks yields worse results than on the reference block, albeit it tends to improve as the *fanout* constraint increases, due to the increase on the clock common path. In terms of *latency*, the best result is obtained when using a *fanout* constraint of 64, where it improves by 3.4% compared to the reference case, being worse in all other cases.

By analysing the number of *repeater*s used, it can be seen that the best results are obtained at a *fanout* constraint of 64 and 128, decreasing by 30% approximately while for a *fanout* constraint of 256 and 512 it only improves by 15% approximately. This indicates that in terms of *repeater* usage, the use of high *fanout* constraints on this block is not suitable.

In terms of power, the best result is obtained when using a *fanout* constraint of 64, decreasing by 3.5% approximately, compared to the 3% improvement obtained at 128. In terms of clock wirelength, similar results are obtained to the ones regarding clock cells. At 64 and 128 *fanout* constraints, the clock wirelength is 6% lower compared to the 3% reduction on the other cases when compared to the reference run.

When comparing the 64 and 128 *fanout* blocks, there are small reductions on dynamic power that could be attributed to differences in the clock routing wirelength, however the 64 *fanout* block has less leakage power consumption, indicating that smaller clock cells are used on average, as the only contributor to the leakage clock power are the *repeater*s and clock gaters used.

These results check out with the previous block where unattainable *fanout* constraints yielded worse results due the algorithms followed by the program. On this block, the result is less pronounced as the block has less memory instances that take less block area.

In terms of general power, the trends followed in the general dynamic power follows the clock power variations while the leakage power is heavily reduced at a *fanout* constraint of 64 by 6%, while on the other cases, less significant reductions are observed.

When considering the clock structure the following results are obtained:

Fanout constraint	32	64	128	256	512
INV_D16	691	688	746	737	1007
INV_D12	2,270	2,086	2,861	3,425	3,433
INV_D6	15,075	10,073	9,187	11,906	10,948

Table 4.11: *Repeater* breakdown distribution at Block 2 for the different *fanout* constraints analysed.

Fanout constraint	32	64	128	256	512
Average *repeater fanout*	14	18	18	16	16
Average *ICG fanout*	9	9	9	9	9
Cells *fanout* =< 16	17,518	15,619	16,190	18,748	18,656
Cells *fanout* => 32	8,156	5,045	4,387	4,423	4,435
Cells *fanout* => 64	0	2,902	1,696	1,608	1,581
Cells *fanout* => 128	0	0	338	332	320
Cells *fanout* => 256	0	0	0	0	0

Table 4.12: *Repeater fanout* breakdown distribution and average clock cells *fanout* at Block 2 for the different *fanout* constraints analysed.

From the results obtained regarding the clock structure it can be seen that:

The average *fanout* increases from 14 to 18 for *fanout* constraints of 64 and 128 while it lower at 256 and 512. The *fanout* distribution indicates that there are no cells using *fanout* constraints larger than 256 despite increasing the maximum *fanout* constraint up to 512.

A low number of cells with a *fanout* constraint of 128 is added when the maximum constraint is raised to that value, compared to the results obtained when raising the *fanout* constraint from 32 to 64 where 2902 cells use the maximum constraint allowed while lowering the number of cells with a *fanout* lower than 16 and bigger than 32.

Overall, when using a *fanout* constraint higher than 128, there are general increases in all the inverter types compared to the other cases.

This indicates degradation on the clock structure for large *fanout* constraints on this block.

The *fanout* constraint of 128 could also be discarded in terms of structure due to the low number of cells using the maximum *fanout* given, compared to the results obtained at lower *fanout*s.

4.4.3. Block 3 Analysis

After the initial *fanout* analysis was performed on the first to blocks it was checked the heavy correlation between the block physical layout and the maximum *fanout* constraint that can be applied without incurring in degradation of the clock structure, and via *CCD* of the datapath.

However, in comparison with the other blocks analysed regarding *fanout*, the characteristics of Block 3 differ greatly.

The lack of memories, and thus blockages, results on an overall smoother clock and data routing with more flexibility on the placement of cells. It is important to note too, that due to the lack of memories, the in-between memory flop placement cannot occur, which resulted on the main cause of *fanout* limitation and worse initial timing conditions.

It was initially planned to perform *fanout* analysis up to 64 as it was marked as the maximum *fanout* providing improvements on Block 2, however as it has been explained, the lack of memories makes it more likely that higher *fanout* constraints can be achieved while providing better results.

Based on this assumption, the analysis has been extended up to a limit constraint of 256 instead of 64 as it was planned at the beginning of the analysis.

Regarding timing metric analysis, the first noticeable result is the initial *Worst Negative Slack* and *Total Negative Slack*. In all blocks analysed, the worst result obtained is -0.002 ns, compared to -0.346 ns of Block 1 which has the worst initial results in terms of timing while for the *Total Negative Slack* is zero due to rounding in all *fanout* cases.

In terms of *skew* and insertion, improved results are obtained at a *fanout* constraint of 128, decreasing by 21% and 3.5% respectively, while on the other cases analysed, increments are observed.

Regarding the use of *repeater*s, less instances are used on each case as the *fanout* increases, ranging from 33% at a constraint of 64 up to 38% when a constraint of 256 is used. This trend is followed in the dynamic clock power reduction, scaling from 2.7% up to 3.1% as the *fanout* increases.

On the other hand, the leakage power varies, giving improvements of 7.1% at a 64 constraint, 6.4% at 128 and 7% at 256. From this results and the *repeater* area, it is likely that the average *repeater* size for a *fanout* constraint is much lower than on 128 and 256 due to the correlation between leakage power and *repeater* size.

The total clock power decreases accordingly to the reductions of the clock power considering that, on this block, compared to other blocks, the clock power takes up to 89% of the total dynamic power.

Considering the reference blocks of all the test blocks, the percentages between clock power, general dynamic power and general power is the following:

Block	Clock Total Power (µW)	Total Dynamic Power (µW)	Total Power (µW)	Clock Power / Total Dynamic Power	Clock Power / Total Power
Block 1	723,969.24	1,170,000	1,440,000	61.82%	50.31%
Block 2	430,313.75	875,000	942,000	49.18%	45.68%
Block 3	425,355.94	474,000	486,000	89.73%	87.52%

Table 4.13: Clock power, total power and clock power to total power distribution for all test blocks at a logic activity factor of 10% and a clock activity factor of 200%.

Block 3 has the highest clock power to total power ratio due to the lack of memories which in turn yields the best improvements to the general metrics when looking for an optimized configuration.

When considering the clock structure analysis the following results are obtained:

Fanout constraint	32	64	128	256
INV_D16	433	378	422	409
INV_D12	1,623	1,210	1,675	1,623
INV_D6	8,513	5,469	4,725	4,486

Table 4.14: *Repeater* breakdown distribution at Block 3 for the different *fanout* constraints analysed.

Fanout constraint	32	64	128	256
Average *repeater fanout*	16	21	21	22
Average *ICG fanout*	13	14	14	14
Cells *fanout* =< 16	15,582	13,935	14,162	13,847
Cells *fanout* >= 32	4,885	3,206	2,706	2,696
Cells *fanout* >= 64	0	1,542	924	918
Cells *fanout* >= 128	0	0	279	269
Cells *fanout* >= 256	0	0	0	0

Table 4.15: *Repeater fanout* breakdown distribution and average clock cells *fanout* at Block 3 for the different *fanout* constraints analysed.

As the maximum *fanout* constraint increases, the average *repeater fanout* increases in all the cases analysed. In terms of the cell breakdown, the number of cells with a *fanout* lower than 16 shows a decreasing trend lowering from 15582 cells up to 13847 for the biggest *fanout* constraint.

For the number of cells with a larger *fanout* of 32, it decreases up to 128, where it stabilizes at 2706 instances, without further improving when increasing the maximum *fanout* to 256.

A similar result is seen when the number of cells with *fanout*s larger than 64, with a reduction on the number of instances from 1542 to 924.

One of the points to note is that the *fanout* constraint of 256 is never achieved at the highest *fanout* constraint run, while the number of cells at 128 is approximately the same to the one obtained with a *fanout* constraint of 128.

From these results, it can be seen that using a *fanout* constraint of 256, does not result on big improvements in terms of cell distribution as its maximum constraint is not used. The possible degradation on the clock structure is not as noticeable as on the other blocks analysed due to the lack of memory blockages in the layout.

4.5. Clock Cell Analysis

The selection of the constellations used for each *repeater* combination has been limited to certain *fanout*s to reduce the length of the experiment.

When using bigger inverters only high *fanout* constraints have been considered, with the addition of the reference *fanout* set to provide a complete constellation.

In a similar fashion, the smaller *repeater* sets have been tested with lower *fanout* constraints, on this case 32 and 64 in all cases. To complete the constellations, it has been added the results at the same *fanout* constraints using the reference *repeater* sets.

The reference parameters of the blocks on these analyses are the following:

- Reference clock *fanout* constraint: 32
- Reference *slew* constraint: 80 ps
- Reference set of *repeater* cells: *INV_D16, INV_D12* and *INV_D6.*
- Application of *Concurrent Clock and Data Optimization*: Active
- Position of clock pin: Reference position

4.5.1. Block 1 analysis

When using bigger *repeater*s, the most relevant results obtained are the following ones:

Regarding timing metrics, the use of bigger clock cells yields a *Worst Negative Slack* and *Total Negative Slack* on average twice as much as in the reference case, increasing from -0.346 ns to -0.662 ns at the reference *fanout* and *slew* constraint. When increasing the *fanout* the results do not further degrade but improve in terms of *Total Negative Slack*, showing a reduction on the number of violating clock paths.

The *skew* is worse in all cases, 0.598 ns at the reference block and increasing up to 0.681ns when a *fanout* constraint of 256 and bigger clock cells are used, showing an increased unbalancing in terms of clock arrival at the different flop.. The *insertion delay* remains approximately the same which indicates a similar maximum downstream capacitance from the clock pin to the furthest away flip-flop.

Regarding the usage of clock cells several results must be remarked. At both *fanout* constraints, when compared with their reference cell counterpart more *repeater* instances are used, 15% more instances with the reference *fanout* and 11% more instances with the 256 *fanout* constraint compared to their reference counterpart.

This causes large clock power increases of 10% on average.To further analyse the increase on dynamic power, it can be seen that the clock wirelength is on average 11% bigger than on the reference case. Part of the dynamic power increase will be related to the larger wirelength, while the other part will be due to the increase of instances and the increase on the individual power consumption per *repeater* due to the increased size.

This result coupled with the increase on the number of instances used, makes it fair to assume that the usage of bigger cells caused unbalance in terms of WNS, and capacitance balancing which forced using more instances and larger clock tracks.

Regarding general power, the increase on the dynamic power is mainly due to the increase on the clock power.

When the clock cell structure is considered the following results are obtained:

Contellation (*fanout*; *repeater* size)	32;16_12_6	256;32_12_6	32;24_16_8	256;24_16_8
INV_D16	2,172	1,203	X	X
INV_D12	3,630	4,013	X	X
INV_D6	23,167	19,131	X	X
INV_D24	X	X	2,013	1,370
IND_D16	X	X	2,018	5,567
INV_D8	X	X	26,441	20,712

Table 4.16: *Repeater* breakdown distribution at Block 1 for the different clock cell and *fanout* constellations using the bigger *repeater* sets.

Contellation (*fanout*; *repeater* size)	32;16_12_6	256;32_12_6	32;24_16_8	256;24_16_8
Average *repeater fanout*	15	17	13	15
Average *ICG fanout*	5	7	7	7
Cells *fanout* =< 16	20,264	20,619	24,108	24,385
Cells *fanout* => 32	11,078	5,920	10,642	5,502
Cells *fanout* => 64	0	2,628	0	2,480
Cells *fanout* => 128	0	45	0	414
Cells *fanout* => 256	0	0	0	0

Table 4.17: *Repeater fanout* breakdown distribution and average clock cells *fanout* at Block 1 for the different clock cell and *fanout* constellations using the bigger *repeater* sets.

From the results in Table 4.17 it can be seen that by increasing the clock cell size the average *repeater fanout* is reduced. The increase on the lower *fanout* cells is due to the delay balancing.

The main contributor on the number of instances used is the smaller *repeater* size of the set. The average *fanout* is also reduced while the low *fanout* nets are increased by 20% approximately when compared to the reference set.

The results obtained on the usage of smaller clock cells are the following ones:

It must be considered that the block using the smaller cells and a *fanout* constraint of 64 has an increase on the number of *DRC*s of 66% compared to the reference set and it is likely to have errors in the development and in the results obtained.

As seen on the previous analysis on bigger cell usage, the *Worst Negative Slack* increases by 60% on average in all the cases analysed.

The *skew* obtained improves by 8% in the set with the reference *fanout* and smaller *repeater*s while it is reduced to 0.7% when increasing the *fanout* to 64.

In both cases analysed when using smaller cells the *insertion delay* is reduced, an expected result on reducing the *repeater* parasitic capacitance.

The number of *repeater*s is reduced when using the reference *fanout* and smaller *repeater*s by 14% compared to the base case while it increases when the *fanout* is increased to 64. This increase is probably due to possible errors obtained on the run.

Regarding clock cell instances used, at a *fanout* constraint of 32, there is a reduction of 15% while for 64, there is a small increase on the number of instances used when comparing the smaller clock cell and the bigger clock cell values.

When considering the clock power, at a *fanout* constraint of 32 there are reductions on the dynamic and leakage power. The clock wirelength at 32 is also lower than on the reference case.

Regarding clock power, at a *fanout* of 32 using smaller cells the dynamic power is reduced by 3.9% while the leakage power is reduced by 33% due to the reduction on the number of *repeater*s used.

In terms of clock structure analysis the following results are obtained:

Contellation (*fanout*; *repeater* size)	32;16_12_6	64;16_12_6	32;14_10_4	64;14_10_4
INV_D16	2,172	1,203	X	X
INV_D12	3,630	4,013	X	X
INV_D6	23,167	19,131	X	X
INV_D14	X	X	1,889	1,325
IND_D10	X	X	3,987	4,699
INV_D4	X	X	18,413	25,370

Table 4.18: *Repeater* breakdown distribution at Block 1 for the different clock cell and *fanout* constellations using the smaller *repeater* sets.

Contellation (*fanout*; *repeater* size)	32;16_12_6	64;16_12_6	32;14_10_4	64;14_10_4
Average *repeater fanout*	15	14	17	13
Average *ICG fanout*	5	7	7	7
Cells *fanout* =< 16	20,264	21,222	14,934	22,072
Cells *fanout* => 32	11,078	10,755	11,042	10,682

Table 4.19: *Repeater fanout* breakdown distribution and average clock cells *fanout* at Block 1 for the different clock cell and *fanout* constellations using the smaller *repeater* sets.

As specified before, the results on the run using a *fanout* of 64 and smaller clock cells is likely to have errors and it should be taken into consideration with the results obtained.

The reduction of the cell capacitance on the inverters used has the following consequences:

The biggest inverters are used more due to the reduced capacitance, similarly to the mid-sized inverters. The reduction on size allows lower *fanout*s to be used and increase the possible *fanout* brackets on each cell.

The smallest inverters have a reduced use as they qualify only for low *fanout*s due to the reduction on the cell capacitance.

4.5.2. Block 2 Analysis

The selection of the constellations has been done following the same reasoning as in Block 1.

The analysis done regarding the usage of bigger clock cells is the following :

The *Worst Negative Slack* on the two test cases analysed increases from -0.06 ns in the reference case to -0.138 ns when the reference *fanout* is used and -0.125 ns when the *fanout* is increased to 256.

The results obtained regarding the *Total Negative Slack* when using a *fanout* of 256 and bigger *repeater*s is 9.7 ns compared to the 3.7 ns obtained on its reference counterpart. This indicates an increase on the number of violating paths from 163 in the reference case to 1089.

As it was obtained previously, the usage of bigger clock cells yields a bigger *skew* due to increased unbalancing on the clock tree structure when using *Concurrent Clock and Data Optimization*. Compared to the reference case, the *skew* increases by 1% when using the reference case, but it is increased by to 37% when the *fanout* constraint is increased to 256.

Similarly, the *insertion delay* increases in both cases and is more noticeable when the *fanout* is increased. When using the reference *fanout*, by using bigger *repeater*s the *insertion delay* increases by 3%, while by using a *fanout* constraint of 256, it increases by 14.5%.

Regarding *repeater* use, both cases using smaller clock cells use less *repeater* instances. On the other hand, the area and power increase due to the increase on the cell size.

The *repeater* area is increased by 32% when the reference *fanout* is used and by 10% when it is increased up to 10%. The total power increases respectively by 3.2% and 1.2%.

In terms of clock wirelength, the run using bigger *repeater*s and the reference *fanout* has a similar *fanout*, increasing only by 1%, while when the *fanout* is increased; the clock wirelength is reduced by 6% compared to the 3% obtained before.

Regarding the clock tree structure, the results obtained are summarized on the following tables:

Contellation (*fanout*; *repeater* size)	32;16_12_6	256;16_12_6	32;24_16_8	256;24_16_8
INV_D16	691	737	X	X
INV_D12	2,270	3,425	X	X
INV_D6	15,075	11,280	X	X
INV_D24	X	X	1,048	930
IND_D16	X	X	2,033	2,579
INV_D8	X	X	14,550	10,215

Table 4.20: *Repeater* breakdown distribution at Block 2 for the different clock cell and *fanout* constellations using the bigger *repeater* sets.

Contellation (*fanout*; *repeater* size)	32;16_12_6	256;16_12_6	32;24_16_8	256;24_16_8
Average *repeater fanout*	14	16	14	17
Average *ICG fanout*	9	9	9	9
Cells *fanout* =< 16	17,518	18,748	17,147	17,506
Cells *fanout* => 32	11,078	4,423	8,360	4,192
Cells *fanout* => 64	0	1,608	0	1,695
Cells *fanout* => 128	0	332	0	501
Cells *fanout* => 256	0	0	0	0

Table 4.21: *Repeater fanout* breakdown distribution and average clock cells *fanout* at Block 2 for the different clock cell and *fanout* constellations using the bigger *repeater* sets.

From the results obtained it can be seen that when using smaller cells, more *INV_D24* cells are used compared to *INV_D16* while less instances are used on the smaller *repeater* cells used.

The usage of bigger clock cells helps increasing the number of nets that can be promoted to higher *fanout* nets. The results are more pronounced at a *fanout* constraint of 256 where the average *repeater fanout* is increased. The bigger clock cells are increased 737 to 930 instances while the smaller inverters have a reduction on 1911 instances.

On this case, the results are the expected, the average *fanout* increases, resulting on a higher number of high *fanout* nets with the reduction on the lower *fanout* side.

The results obtained when using the smaller set of *repeater*s is the following:

On the test using the reference *fanout* it has been observed an increase on the number of *DRC* violations of 682% compared to the reference case and thus will not be included on this analysis.

The *Worst Negative Slack* in both cases analysed when using smaller clock cells is worse. From the reference value of -0.06 ns it increases up to -0.131 ns when a *fanout* of 64 is used.

Regarding *skew*, it is reduced when the smaller inverters are used, lowering from 0.206 ns to 0.204ns. The *insertion delay* is increased by 1.44% compared to the reference case.

The number of *repeater*s used on the block using a *fanout* constraint of 64 is reduced by 20%. This results on a reduction on the clock dynamic and leakage power by 3.4% and 18% respectively.

In terms of general power, the dynamic power is reduced by 1.37% mainly due to the reductions on the clock power. The reduction on the number of *repeater*s used is due to the increase on the *fanout* constraint.

Regarding the clock cell structure, the results obtained are the following ones:

Contellation (*fanout*; *repeater* size)	32;16_12_6	64;16_12_6	32;14_10_4	64;14_10_4
INV_D16	691	688	X	X
INV_D12	2,270	2,086	X	X
INV_D6	15,075	10,073	X	X
INV_D14	X	X	1,155	1,340
IND_D10	X	X	3,625	4,479
INV_D4	X	X	12,763	8,610

Table 4.22: *Repeater* breakdown distribution at Block 2 for the different clock cell and *fanout* constellations using the smaller *repeater* sets.

Contellation (*fanout*; *repeater* size)	32;16_12_6	64;16_12_6	32;14_10_4	64;14_10_4
Average *repeater fanout*	14	18	14	17
Average *ICG fanout*	9	9	9	9
Cells *fanout* =< 16	17,518	15,619	16,936	16,607
Cells *fanout* => 32	11,078	5,045	8,213	5,113
Cells *fanout* => 64	0	2,902	0	2,261

Table 4.23: *Repeater fanout* breakdown distribution and average clock cells *fanout* at Block 2 for the different clock cell and *fanout* constellations using the smaller *repeater* sets.

When comparing the usage of clock cells it can be seen that when using the set with smaller cells, there are more instances using the bigger cells as it will allow for lower *fanout*s compared to the use of the cells in the reference set. The usage of the smaller *repeater*s in the smaller set of cells also decreases as some nets will be using bigger cells leaving only the really low *fanout* nets.

By analysing the *repeater* breakdown it can be seen that overall the average *fanout* decreases, as it should be expected of using sets with lower driving power.

4.5.3. Block 3 Analysis

The selection of constellations regarding the selection of clock cells in Block 1 and Block 2 have been done considering that high *fanout* constraint were achievable.

As it has been seen on the analysis done regarding clock cells and *fanout* and clock cell selection in Block 1 and Block 2, the clock building algorithm tends to use the biggest *fanout* available even though it yields sub-par results.

This limited the usefulness of using different size clock cells due to the following reasons:

- A lower effective maximum *fanout* constraint limit due to area restrictions made the usage of bigger *repeater*s unnecessary.
- The initial clock structure was limited by the placement of memories resulting on increased *Worst Negative Slack*.

On Block 3, however there are no area restrictions due to the absence of memories. As seen on the *fanout* analysis, it is possible to achieve *fanout* constraints up to 256 without severe degradation on the metrics obtained.

Taking into account the results obtained in the *fanout* analysis, the set of constellations defined on the experiment sets for Block 3 have been used.

Regarding the usage of bigger clock cells, the results obtained are the following ones:

The *Worst Negative Slack* is worse than on the reference block, however, the initial results regarding timing makes it not critical.

The *skew* in all the cases analysed by using bigger clock cells improves up to 45% compared to the reference block, when the *fanout* is increased to 256. Similar results are obtained in the *insertion delay*, where the *fanout* constraint of 256 run yields the best result, improving by 14% compared to the reference run.

The number of *repeater*s used follows a similar trend to the one obtained when using the reference cells, improving by 33% when a *fanout* constraint of 128 is used and by 41% when a *fanout* constraint of 256 is used.

The clock power improvements are more modest due to the increase on the bigger clock cells, reducing the clock power by 1.3% and 1.9% respectively compared to its reference *repeater* size counterpart where it sports reductions of 2.8% and 3.1%.

In terms of general power, the total power is reduced on both cases mainly due to the

The clock tree structure results obtained are the following ones:

Contellation (*fanout*; *repeater* size)	32;16_12_6	128;16_12_6	256;16_12_6	32;24_16_8	128;24_16_8	256;24_16_8
INV_D16	433	422	409	X	X	X
INV_D12	1,623	1,675	1,623	X	X	X
INV_D6	8,513	4,725	4,486	X	X	X
INV_D24	X	X	X	477	409	369
IND_D16	X	X	X	1,116	1,303	1,305
INV_D8	X	X	X	8,595	5,374	4,544

Table 4.24: *Repeater* breakdown distribution at Block 3 for the different clock cell and *fanout* constellations using the bigger *repeater* sets.

Contellation (*fanout*; *repeater* size)	32;16_12_6	128;16_12_6	256;16_12_6	32;24_16_8	128;24_16_8	256;24_16_8
Average *repeater fanout*	16	21	22	17	21	23
Average *ICG fanout*	13	14	14	13	14	14
Cells *fanout* =< 16	15,582	14,162	13,847	15,181	14,630	13,678
Cells *fanout* => 32	4,885	2,706	2,696	4,970	2,711	2,626
Cells *fanout* => 64	0	924	918	0	881	875
Cells *fanout* => 128	0	279	269	0	308	408
Cells *fanout* => 256	0	0	0	0	0	0

Table 4.25: *Repeater fanout* breakdown distribution and average clock cells *fanout* at Block 3 for the different clock cell and *fanout* constellations using the bigger *repeater* sets.

When analysing the clock structure it can be seen that when using bigger clock cells, the inverter *INV_D24* and *INV_D16* are less used to its counterpart on the reference clock cell set, *INV_D16 and INV_D12*, as the increase on driving power will make the cells be selected only for high *fanout* nets. On the other hand, the smallest cell, *INV_D8*, is used in the low *fanout* nets as well as some higher nets that do not use *INV_D12*.

Overall, considering the cell breakdown, the average *fanout* tends to increase at high *fanout* constraints giving a better use to the cell set using bigger inverters, as it can be seen on the *fanout* distribution, where the number of cells with a *fanout* greater of 128 increase while having slight decreases for low and mid *fanout* cells.

On the other hand, the results when using smaller clock cells are the following ones:

Similar to the results obtained when using bigger clock cells, there is a slight degradation regarding *Worst Negative Slack* but it is not critical on the design.

The *skew* obtained is improved by 11.1% and 25% when setting the *fanout* constraint to 32 and 64 respectively and smaller clock cells are used. The *insertion delay* in both cases improves by 4.2% and 5.81% respectively, as expected from the reduction on the clock cell parasitic capacitance by lowering its size.

The reduction on the number of instances used follows a similar trend to the obtained when using the reference clock cells, decreasing by 3.4% with the reference clock cells and by 30% when the *fanout* is increased to 64.

Regarding clock power, the run using the reference *fanout* and smaller *repeater*s have a 1.6% reduced total power, while it sports a reduction of 2.4% when the *fanout* is increased to 64.

In terms of general power, the clock dynamic power reduction causes the reduction on the overall dynamic power while the leakage power varies in both cases. Overall, in both cases the total power is reduced, mainly due to the reduction in the clock power.

The clock structure information obtained is the following one:

Contellation (*fanout*; *repeater* size)	32;16_12_6	64;16_12_6	32;14_10_4	64;14_10_4
INV_D16	433	378	X	X
INV_D12	1623	1210	X	X
INV_D6	8513	5469	X	X
INV_D14	X	X	464	555
IND_D10	X	X	2177	2251
INV_D4	X	X	7569	4775

Table 4.26: *Repeater* breakdown distribution at Block 3 for the different clock cell and *fanout* constellations using the smaller *repeater* sets.

Contellation (*fanout*; *repeater* size)	32;16_12_6	64;16_12_6	32;14_10_6	64;14_10_4
Average *repeater fanout*	16	21	17	19
Average *ICG fanout*	13	14	13	14
Cells *fanout* =< 16	15,582	13,935	15,181	14,620
Cells *fanout* => 32	4,885	3,206	4,879	3,216
Cells *fanout* => 64	0	1,542	0	1,328

Table 4.27: *Repeater fanout* breakdown distribution and average clock cells *fanout* at Block 3 for the different clock cell and *fanout* constellations using the smaller *repeater* sets.

The reduction on the *repeater* size, and thus driving power will result on less high *fanout repeater*s, while the nets with lower *fanout* increase. This lowers the average *fanout* obtained when a *fanout* of 64 is used, being reduced from 21 to 19.

When considering the inverter usage, the biggest inverters, *INV_D14* and *INV_D10,* are more used compared to its reference cells counterpart with a reduction on the usage on the smallest *repeater*s. This result is as expected as the biggest inverters will qualify for a larger *fanout* bracket while the smallest cells will only be used for smaller *fanout*s, compared to their reference counterpart.

The last analysis that will be covered in Block 3 was due to an error obtained on the definition of the clock cells. On this case, the reference and the smaller *repeater* set was used. The most relevant results obtained on this case are referred to the clock tree structure.

The results obtained are summarized on the following tables:

Contellation (*fanout*; *repeater* size)	64;16_12_6	64;mixed_sizes
INV_D16	378	137
INV_D12	1,210	169
INV_D6	5,469	2,730
INV_D14	X	385
IND_D10	X	1,678
INV_D4	X	2,241

Table 4.28: *Repeater* breakdown distribution at Block 3 for the different clock cell and *fanout* constellations using a mixed set.

Contellation (*fanout*; *repeater* size)	64;16_12_6	64;mixed_sizes
Average *repeater fanout*	21	20
Average *ICG fanout*	14	14
Cells *fanout* =< 16	13,935	14,096
Cells *fanout* => 32	3,206	3,218
Cells *fanout* => 64	1,542	1,438

Table 4.29: *Repeater fanout* breakdown distribution and average clock cells *fanout* at Block 3 for the different clock cell and *fanout* constellations using a mixed set.

From the results obtained it can be seen that:

When using 6 clock cells, two of the *repeater*s are not selected favouring others.

- *INV_D16* has 137 instances compared to the 378 instances of *INV_D14.*
- *INV_D12* has 169 instances compared to the 1678 instances of *INV_D10.*

On the last pair of inverters, the results are more balanced, although *INV_D6* is favoured over *INV_D4*.

Taking this into consideration, the usage of multiple *repeater* sets is not useful nor justified as some *repeater* sizes will be favoured over other that will have a low utilization ratio.

4.6. Concurrent Clock and Data Optimization Analysis

The analysis regarding *Concurrent Clock and Data Optimization* has only been performed in Block 1 and Block 2. The result analysis in those blocks was decisive enough to not perform the analysis in Block 3.

The reference parameters of the blocks on these analyses are the following:

- Reference clock *fanout* constraint: 32
- Reference *slew* constraint: 80 ps
- Reference set of *repeater* cells: *INV_D16, INV_D12* and *INV_D6.*
- Application of *Concurrent Clock and Data Optimization*: Active
- Position of clock pin: Reference position

The different tests used have the following parameter variations:

- *Concurrent Clock and Data Optimization* not applied with *Synopsys* clock routing command and *local skew* optimization.
- *Concurrent Clock and Data Optimization* not applied using a custom clock tree building command set and *local skew* optimization.
- *Concurrent Clock and Data Optimization* not applied with *Synopsys* clock routing command without *local skew* optimization.

4.6.1. Block 1

The most relevant results regarding the *Quality of Results* at Block 1 are the following ones:

The *Worst Negative Slack* and *Total Negative Slack* are much worse than the results obtained on the reference run. The degradation is due to both not using *Concurrent Clock and Data Optimization* and the physical layout of the block. By having blockages due to memory placement, the clock routing is much more limited and the base metrics become much worse by default.

The *Total Negative Slack* increases, shows an increase on the number of violating paths from 10243 to 17585 in the worst case observed when the *CCD* application options are disabled and *local skew* is not used.

By disabling *Concurrent Clock and Data Optimization* options, the *skew* improves by up to 77% when the routing command and the *local skew* configuration options are applied. This indicates the change on the clock tree building a zero-*skew* algorithm.

Regarding the number of *repeater*s used, in all cases there is a reduction on the number of cells used by 16.7% on the best of cases.

This reduction on the number of clock cells used, results on a reduction of 1.2% on average on the total power. On the other hand, the total power increases by 4% on average mainly driven by increases on the total leakage power.

The clock tree structure with a zero-*skew* balancing strategy will require a simpler structure with fewer *repeater*s. The reduction on the number of clock cells used on clockpaths will result on better *insertion delay* and the *skew* improvement is related to the clock tree building strategy followed.

Due to the reduction on the number of *repeater*s used and the reduced clock wirelength, both the clock dynamic and leakage power are lower than on the reference design with *Concurrent Clock and Data Optimization* activated.

On the other hand, the general dynamic and leakage power suffer increases from the lack of datapath optimization that was performed with *Concurrent Clock and Data Optimization*.

When the clock structure is analysed the following results are obtained:

Block reference	noccd_reference_block	noccd_route_local	noccd _custom_local	noccd _route_nolocal
INV_D16	2,172	2,096	785	914
INV_D12	3,630	4,789	6,421	3,824
INV_D6	23,167	17,254	18,116	21,038

Table 4.30: *Repeater* breakdown distribution at Block 1 for the different *CCD* block variations analysed.

Block reference	noccd_reference_block	noccd_route_local	noccd _custom_local	noccd _route_nolocal
Average *repeater fanout*	15	17	17	16
Average *ICG fanout*	5	7	7	7
Cells *fanout* =< 16	20,264	14,833	16,066	16,509
Cells *fanout* => 32	11,078	11,515	11,515	11,513

Table 4.31: *Repeater fanout* breakdown distribution and average clock cells *fanout* at Block 1 for the different for the different *CCD* block variations analysed.

From the results obtained it can be seen that overall, when *CCD* is deactivated the number of instances used are reduced, having the most noticeable reductions for the biggest *repeater*s used when a custom tree is used. In all cases, the usage of the smallest inverter is reduced.

This can be seen with the average *repeater fanout* which increases from 15 to 17 when local optimization is applied and the overall reduction of the number of cells with a *fanout* lower than 16 while larger *fanout* cells are kept more or less constant.

4.6.2. Block 2

The results regarding *QoR* obtained on Block 2 are akin to the results obtained in Block 1. In all the cases where *Concurrent Clock and Data Optimization* configuration options were deactivated, the *Worst Negative Slack* and *Total Negative Slack* results were degraded.

Similarly, the *skew* improved due to the variation on the clock building strategy up to 40% when the routing command is used and the *local skew* optimization is enabled. On the other hand, the *insertion delay* is worse when *local skew* optimization is applied.

A reduction on the number of instances is seen when a custom tree algorithm is used and local optimization is enabled and when the routing command is used without local optimization.

Regarding clock power, only when the *local skew* is disabled improvements in clock power are achieved, reducing by 1.3% compared to the other cases where increases on the clock power of 1.94% and 0.24% are observed.

On the same line as in Block 1, the leakage increases by 15% on average while the dynamic power variations are mostly due to the clock power variations.

The clock structure data obtained is the following one:

Block reference	noccd_reference_block	noccd_route_local	noccd _custom_local	noccd _route_nolocal
INV_D16	691	1,466	494	980
INV_D12	2,270	2,808	5,092	1,446
INV_D6	15,075	12,771	13,844	14,032

Table 4.32: *Repeater* breakdown distribution at Block 2 for the different *CCD* block variations analysed.

Block type	noccd_reference_block	noccd_route_local	noccd _custom_local	noccd _route_nolocal
Average *repeater fanout*	14	13	13	15
Average *ICG fanout*	9	9	9	9
Cells *fanout* =< 16	20,264	16,487	18,922	15,934
Cells *fanout* => 32	11,078	8,364	8,335	8,363

Table 4.33: *Repeater fanout* breakdown distribution and average clock cells *fanout* at Block 2 for the different for the different *CCD* block variations analysed.

Regarding the clock instances used, it can be seen that the best results obtained are in block using the *Synopsys* clock routing command without clock optimization. This can be seen by the increase on the average *repeater fanout*.

When considering the *fanout* breakdown, the results follow the expected trend from the change of structure where the number of end-point *repeater*s decrease and their *fanout* increases compared to the reference run.

4.7. Clock Placement Analysis

As defined before, the experiment set is based around centring the clock pin in order to improve metrics such as the *insertion delay* and *skew* with *CCD* active in order to not degrade the *Worst Negative Slack* results as it has been seen in the previous section.

4.7.1. Blocks Clock Tree Trunk

This initial analysis regarding clock pin position has been done under the following conditions:

The reference parameters of the blocks on these analyses are the following:

- Reference clock *fanout* constraint: 32
- Reference *slew* constraint: 80 ps
- Reference set of *repeater* cells: *INV_D16, INV_D12* and *INV_D6.*
- Application of *Concurrent Clock and Data Optimization*: Active

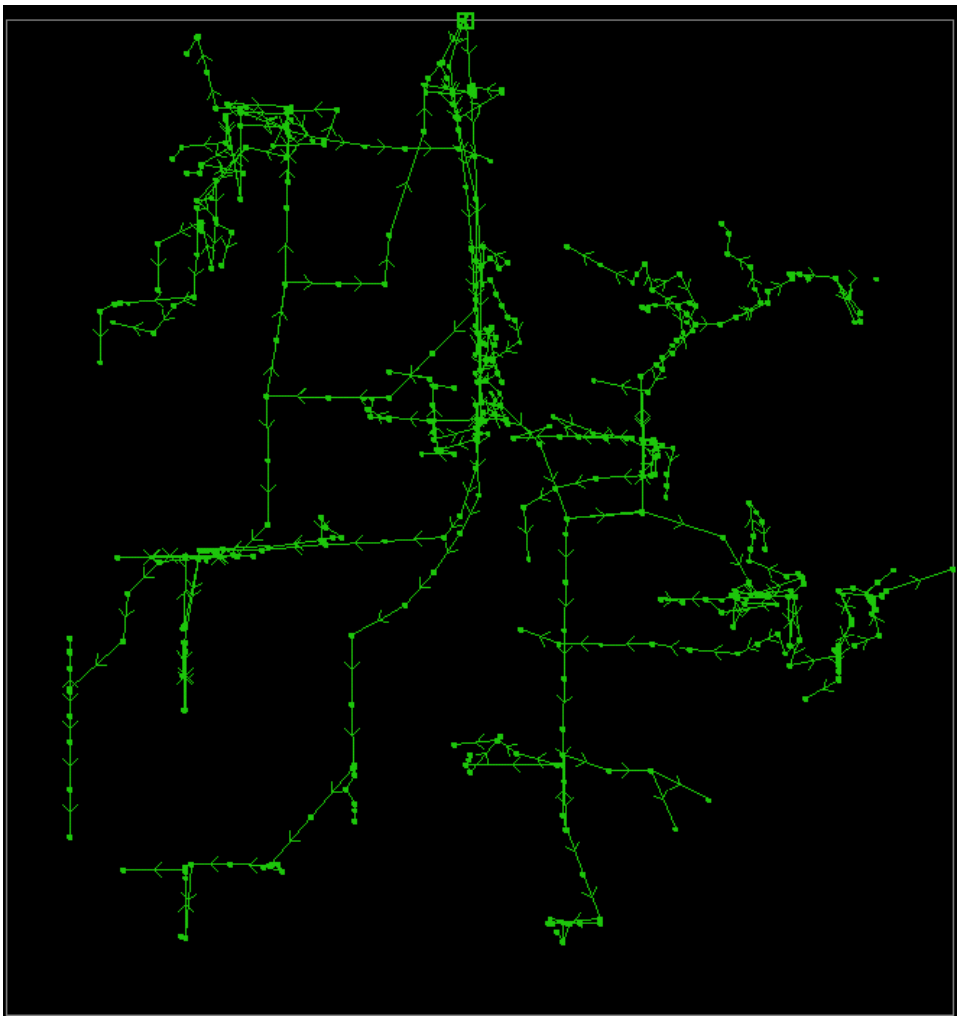

Figure 4.1: *Clock tree trunk of Block 1 using the reference clock tree input pin with CCD algorithm.*

In Block 1, it can be seen that the clock tree propagation algorithm traces different branches on which they are balanced regarding *Worst Negative Slack* minimization. This causes the clock tree to not be balanced around a zero-*skew* structure.

When the clock input pin is changed it can be seen that the structure is simplified in most clock branches.

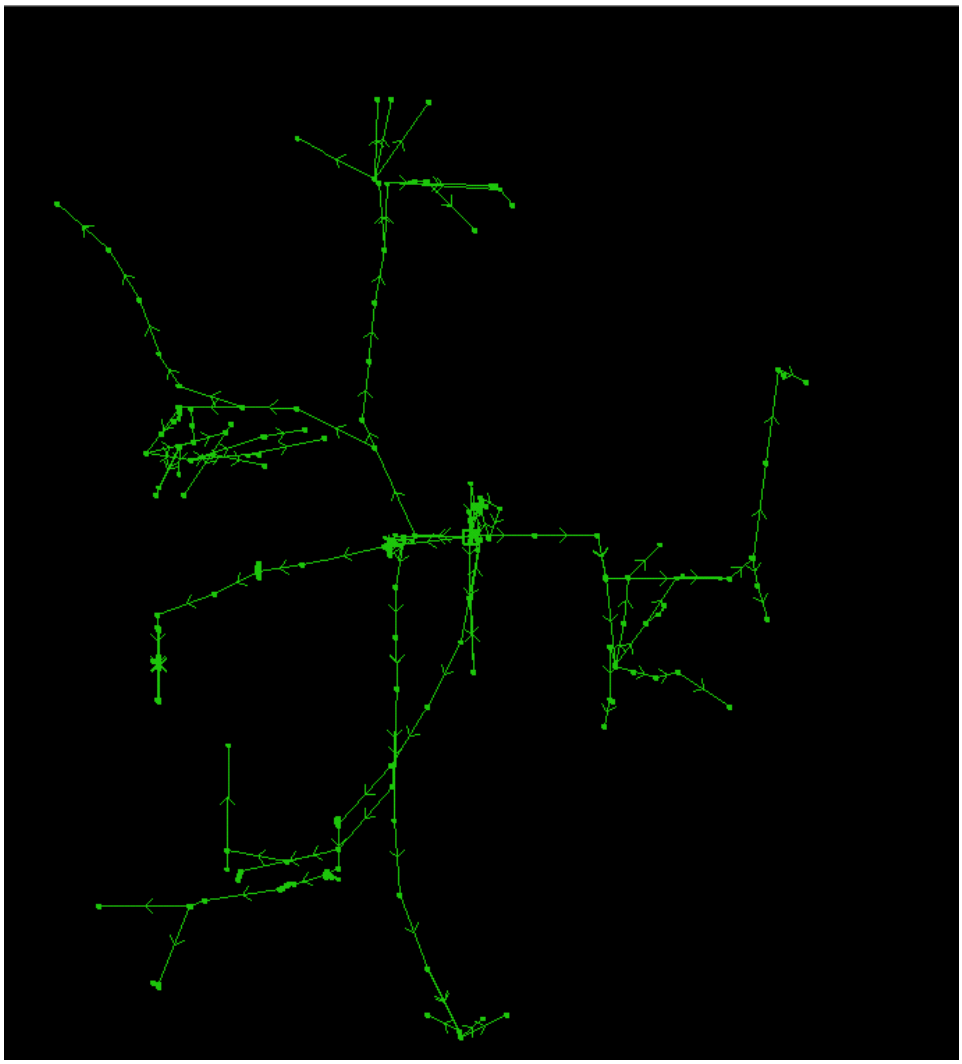

Figure 4.2: *Clock tree trunk of Block 1 using the centred clock tree input pin with CCD algorithm.*

Figure 4.3: *Clock tree trunk of Block 2 using the reference clock tree input pin with CCD algorithm.*

When comparing the structures in block 2, the clock structure obtained is worse than the results obtained in terms of clock branches and clock trunk wirelength.

Similarly to the previous case, the application of *Concurrent Clock and Data Optimization* requires correlation between the datapath and the clockpath. This is reflected on the clock tree where the clock branches of the different clockpaths are treated independently to minimize the *Worst Negative Slack*.

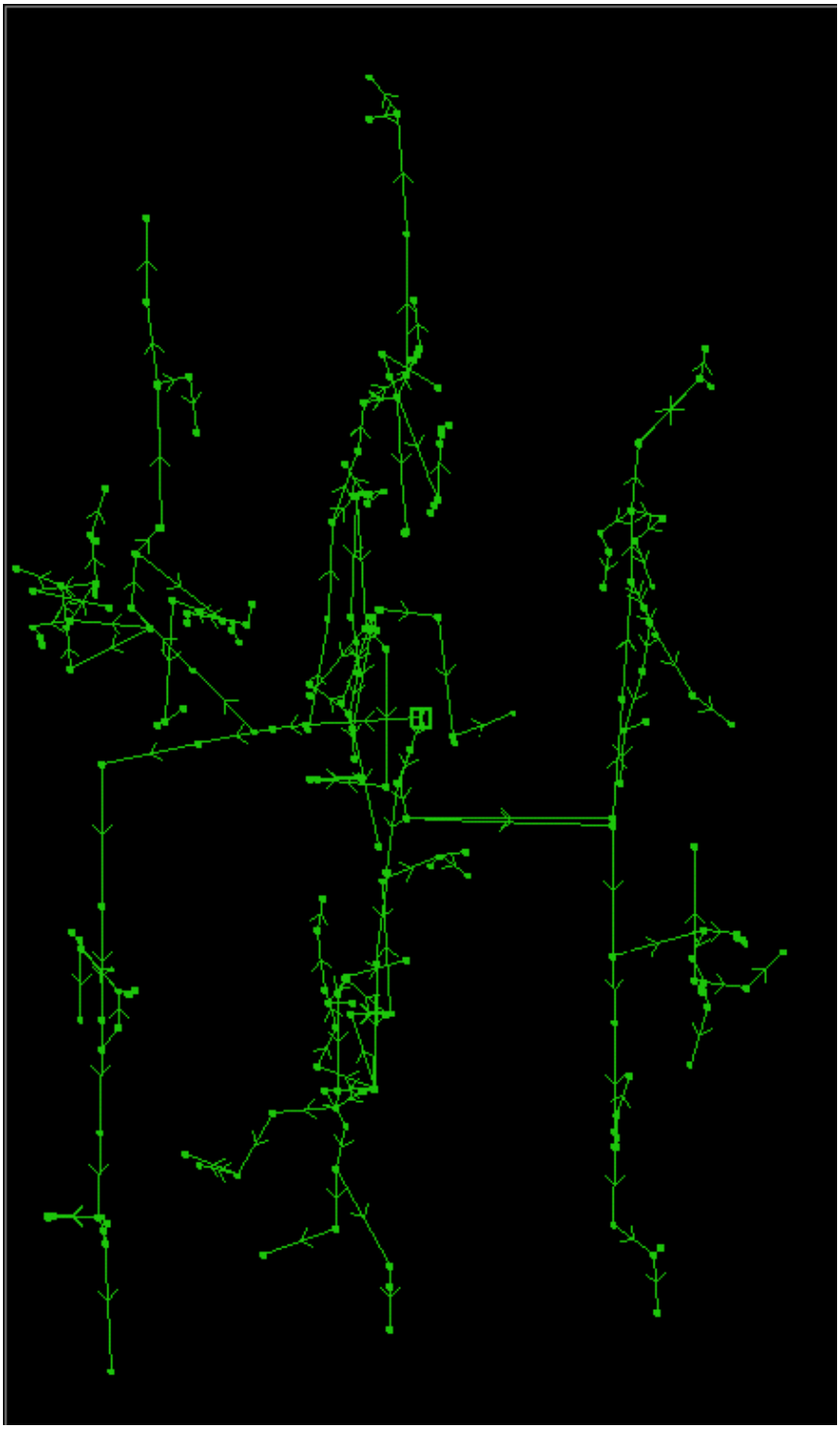

Figure 4.4: *Clock tree trunk of Block 2 using the centred clock tree input pin with CCD algorithm.*

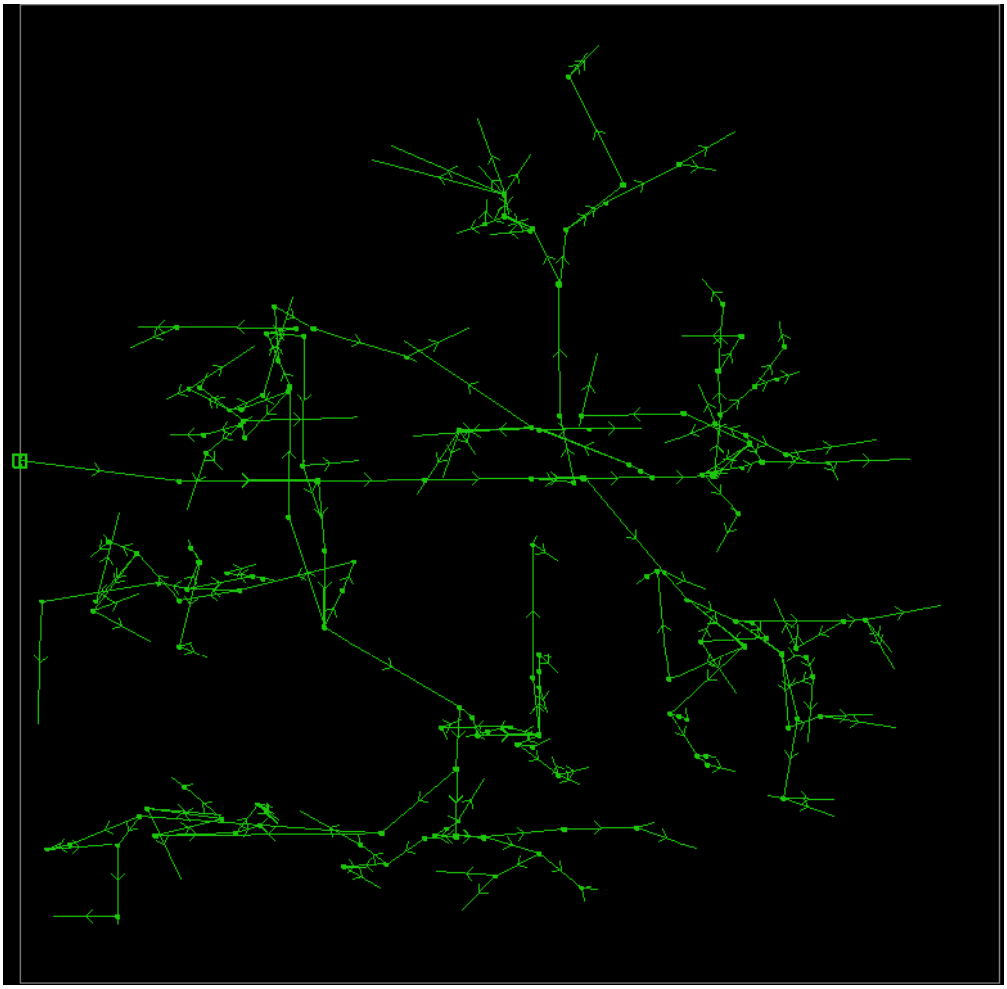

Figure 4.5: *Clock tree trunk of Block 3 using the reference clock tree input pin with CCD algorithm.*

In Block 3, the structure does not vary much by centring the clock tree as the physical layout had no memories.

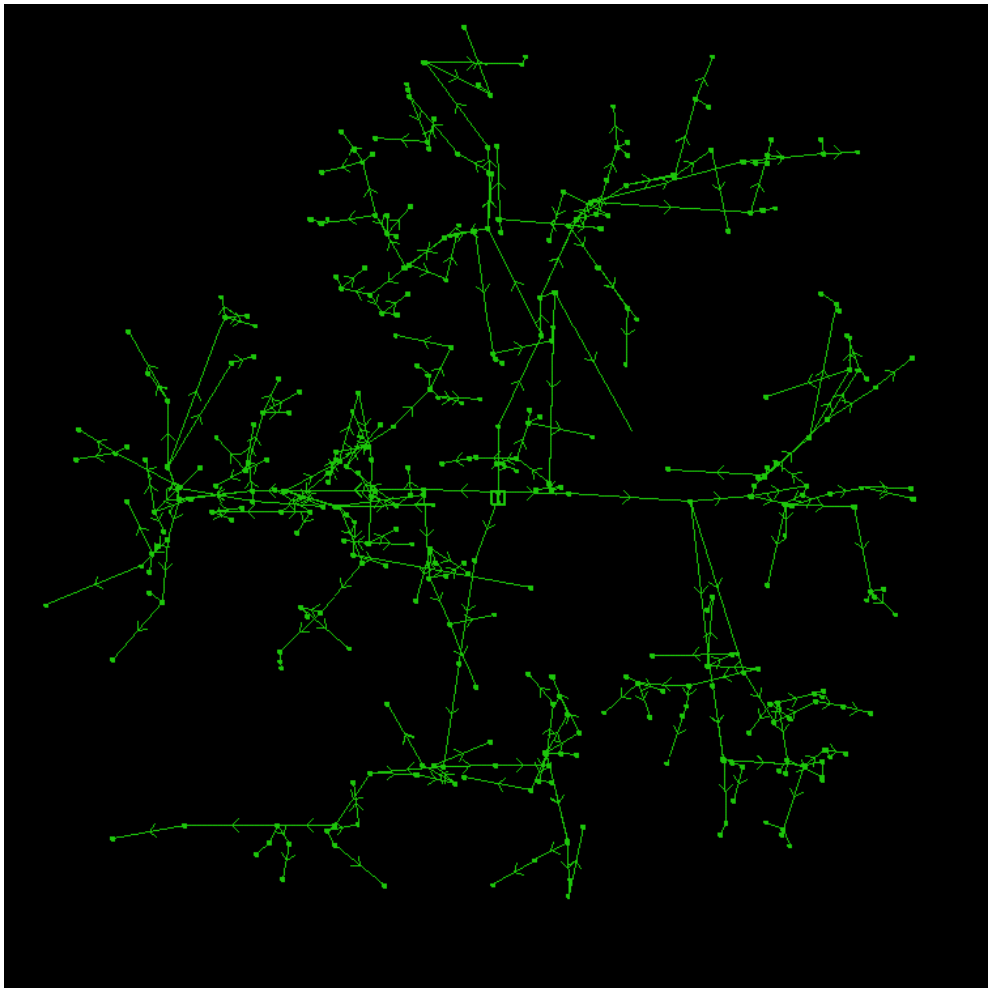

Figure 4.6: *Clock tree trunk of Block 3 using the centred clock tree input pin with CCD algorithm.*

4.7.2. Block 1 Analysis

For Block 1 it has only been tested the parameter combination regarding *fanout* and clock cell selection that gave the best results, that is:

- *Fanout* constraint: 32
- *Repeater* size: *INV_D16, INV_D12* and *INV_D6*

From the structure, it should be expected that the *latency* would improve while the *skew* will depend on *Concurrent Clock and Data Optimization* algorithm.

On the case analysed, it has been obtained an increase on the number of *DRC* violations by 50% and thus it should be taken into consideration when the results are analysed.

Regarding the metrics obtained it can be seen that:

The *Worst Negative Slack* suffers from a slight degradation increasing by 13%, compared to a 33% increase of the *Total Negative Slack*.

By changing the position of the clock pin, the *latency* is improved by 21% due to the change of the clock pin position.

The number of *repeater*s sports an increase of 6% and the clock wirelength increases by 10%.

The clock dynamic power increases by 3.7% and the leakge power increases by 2.68%. Regarding general power, the total dynamic power increases is mainly due to the increase of the clock power while the leakage power increases by 14%

The results obtained in the clock structure are:

Block reference	reference_block	centred_clockpin
INV_D16	2,172	1,242
INV_D12	3,630	4,600
INV_D6	23,167	24,841

Table 4.34: *Repeater* breakdown distribution at Block 1 for the different clock pin configurations analysed.

Block reference	reference_block	centred_clockpin
Average *repeater fanout*	15	14
Average *ICG fanout*	5	7
Cells *fanout* =< 16	20,264	21,343
Cells *fanout* => 32	11,078	10,719

Table 4.35: *Repeater fanout* breakdown distribution and average clock cells *fanout* at Block 1 for the different clock pin configurations analysed.

The biggest *repeater*s decrease, which is expected from the simplification of the clock tree structure, on the other hand, the smallest *repeater*s increase from 3630 to 4600 for *INV_D12* and from 23167 to 24841 for *INV_D6.*

The additional *repeater*s added result on a reduction on the average *repeater fanout*.

4.7.3. Block 2 Analysis

From the experiment sets defined, the reference block with the modified clock pin source gave problems at route optimization and thus it will not be considered at the analysis.

The parameters of the block analysed is the following one:

- *Fanout* constraint: 64
- *Repeater* size: *INV_D16, INV_D12* and *INV_D6*

It should be considered that this run may contain errors as the maximum *fanout* constraint is not achieved, as specified in Table 4.35

In terms of *Worst Negative Slack*, degrades from -0.06 ns to -0.121 ns. Similar to Block 1 and as expected due to the change on the position of the clock pin it improves by 17%.

The number of clock cells used and thus the dynamic and leakage power are worse than its reference clock pin placement block counterpart.

Regarding *repeater* usage, the number of *repeater*s is reduced by 1.2%. The dynamic power lowers by 0.11% while the leakge power increases by 1%. The increase on the leakage power is mainly due to an increase on the average *repeater* size as it can be seen in Table 4.34.

The clock wirelength on the case where the clock pin was moved is larger than on the base case and about 6% bigger than on its default clock pin placement.

Regarding the clock structure information the results obtained are the following ones:

Constellation	32	64	64_moved_pin
INV_D16	691	688	898
INV_D12	2,270	2,086	2,365
INV_D6	15,075	10,073	14,561
AND2_D1	5,880	5,880	5,880

Table 4.36: *Repeater* breakdown distribution at Block 2 for the different clock pin configurations analysed.

Constellation	32	64	64_moved_pin
Average *repeater fanout*	14	18	14
Average *ICG fanout*	9	9	9
Cells *fanout* =< 16	17,518	15,619	17,276
Cells *fanout* => 32	8,156	5,045	8,149
Cells *fanout* => 64	0	2,902	0

Table 4.37: *Repeater fanout* breakdown distribution and average clock cells *fanout* at Block 2 for the different clock pin configurations analysed

When considering the *repeater* usage when the clock pin is moved, the biggest increases obtained are in *INV_D6* compared to its reference placement counterpart. This increase should be considered as a result of the maximum *fanout* achieved on this case. Taking this result into consideration, the results obtained are similar to the ones obtained at the reference block and should probably not be taken into consideration.

4.7.4. Block 3 Analysis

The variations considered on clock pin movement from this block are:

- Reference run with moved pin:
  a. *Fanout* constraint: 32
  b. *Slew* constraint: 80 ps
  c. *Repeater* size: *INV_D16, INV_D12* and *INV_D6.*
- *Fanout* constraint of 128 run with moved pin:
  a. *Fanout* constraint: 128
  b. *Slew* constraint: 80 ps
  c. *Repeater* size: *INV_D16, INV_D12* and *INV_D6.*
- *Fanout* constraint of 128 run with modified *slew* with moved pin:
  a. *Fanout* constraint: 128
  b. *Slew* constraint: 100 ps
  c. *Repeater* size: *INV_D16, INV_D12* and *INV_D6.*

The second experiment set defined did not achieve the defined *fanout* of 128 and was restricted to the reference *fanout* of 32 as it can be seen on Table 4.37, and thus the results obtained could contain errors.

The third variation analysed is focused around the best results obtained in the analysis done previously.

Regarding the *Quality of Results* metrics obtained the results are the following:

The results obtained regarding *Worst Negative Slack* are not relevant as the block has no problems in timing closure.

As in the other blocks analysed, the *latency* improves over the reference block by 18% on the best cases.

The number of *repeater* cells used increases on both cases, having the worse results when using a *fanout* constraint of 128 with a 37% increase. In terms of power, the same trends as in the *repeater* usage with worse power results when comparing the blocks with the initial and the moved pin placement.

In terms of general power, the dynamic power is modified mainly by the clock power. It must be noted that the leakage power shows a decrease compared to all the other cases analysed on this experiment set.

The last case analysed includes the best sets obtained in Block 3, with a *fanout* constraint of 128 and a *slew* constraint of 100 ps. The most relevant results obtained are:

The *insertion delay* improves by 10%. On the other hand, the number of clock *repeater*s is reduced by 55% compared to the 33% improvement of the set using a *fanout* constraint of 128 with the reference clock pin.

This reduction on the number of *repeater*s results on a reduction on the dynamic and leakage power with an overall clock power reduction of 3.2%.

The total power is also reduced mainly due to the reduction on the clock power, while the *DRC* violations are reduced by 20%.

The clock tree structure information obtained is summarized in the following table:

Constellation	32	128	32_moved_pin	128_moved_pin	128_*slew*100_moved_pin
INV_D16	433	422	441	486	274
INV_D12	1,623	1,675	1,436	1,559	1,328
INV_D6	8,513	4,725	8,436	8,791	3,060

Table 4.38: *Repeater* breakdown distribution at Block 3 for the different clock pin configurations analysed.

Constellation	32	128	32_moved_pin	128_moved_pin	128_*slew*100_moved_pin
Average *repeater fanout*	16	21	16	16	30
Average *ICG fanout*	13	14	13	13	14
Cells *fanout* =< 16	15,582	14,162	15,309	15,832	11,606
Cells *fanout* >= 32	4,885	2,706	4,892	4,837	2,722
Cells *fanout* >= 64	0	924	0	0	953
Cells *fanout* >= 128	0	279	0	0	303
Cells *fanout* >= 256	0	0	0	0	0

Table 4.39: *Repeater fanout* breakdown distribution and average clock cells *fanout* at Block 3 for the different clock pin configurations analysed.

From the last analysis performed, where the *slew* and *fanout* were modified it can be seen that the average *repeater fanout* increases from 21 to 30, resulting on a reduction on the number of cells with a *fanout* lower than 16. On the other hand, the biggest *fanout* cells increase.

Regarding the clock cell distribution , on the last analysis performed, there is an overall reduction on all the inverter sizes used, probably mostly due to the variation on the *fanout* constraint.

5. Budget

Due to the nature of the project, the budget will only include the costs referring to the personal hour works.

The dedication of this project has been of 1050 hours. Considering a salary of 15€ an hour as an estimation of a junior engineering in Physical Design, the personal cost would amount to:

$$\text{Total Cost: } 1050 \text{ hours} \cdot 15€/\text{hour} = 15750€$$

It will not be covered the cost of the software and licenses used during the development of this project.

6. Conclusions and future development

This section will include the conclusions and optimal results for each type of block considering the analysis done previously. It will be covered all the experiment sets but the initial set which will be omitted for the reasons given previously.

6.1. Slew Analysis

As explained in chapter 3, the slew constraint is selected in order to minimize variability and clock jutter and it correlates with the separations between repeaters in the clock structure.

This becomes even more important when multiple scenarios are considered where power supply voltage and temperature variations are checked.

From the results obtained several conclusions can be obtained:

- Using a larger *slew* constraint than the optimal one will result on an increase on the number of violating paths due to *repeater*s having *slew* violations.
- Using a smaller *slew* constraint than the optimal one will result in an increase on the number of *repeater*s used on the design.

Typically, when using a smaller *slew* constraint will result on an increase of the smallest end-point *repeater*s while the *ICG fanout* remains mostly constant as it is fixed in the design.

In Block 1 and Block 2, further reductions on the slew constraint did not provide improvements on the quality of results obtained but rather resulted on increases on the number of clock instances and clock power.

From the results obtained, the optimal *slew* constraints for the different blocks are:

- Block 1: 80ps
- Block 2: 80ps
- Block 3: 100ps

6.2. Fanout Analysis

The selection of the *fanout* constraint is vital in terms of obtaining the best result in an optimization process. From the tests made it is not possible to obtain a *fanout* constraint that suits all blocks giving the best result possible as it has been seen.

The physical layout of the block is the main conditioner on the selection of *fanout*. When considering timing metrics, the usage of *Concurrent Clock and Data Optimization* has to be considered as to how the metrics can evolve, and the indeterminacy it adds compared to zero-*skew* balanced structures.

The placement of memories has several consequences that have to be considered.

- Usage of memories reduces the non-blocked space present on the block to place cells and perform routing.
- Placing memories close with small distances between them.
- Placing flip-flops in-between memories will further degrade the results obtained in terms of metrics.

6.2.1. Block 1

The usage of a high number of memory instances placed all across the block with flip-flops placed in-between the memories limits the maximum *fanout* attainable.

For blocks with high memory density the *fanout* should be kept at a low *fanout* constraint to avoid clock structure degradation.

On this particular case, a *fanout* of 32 has been selected as further increasing causes and overall *QoR* degradation. Lowering the clock *fanout* constraint can actually be detrimental as the number of clock cells used would increase.

6.2.2. Block 2

On this case, the physical layout contains less memory instances. Coupled with a reduced memory density with less memory instances and an improved flop placement, the base metric results are better.

On the particular case analysed and extended to blocks with similar physical layout characteristics, the *fanout* chosen was 64, up from the reference *fanout* of 32. Higher *fanout* values cause clock tree degradation due to the clock tree building algorithm.

6.2.3. Block 3

The physical layout on this case is the extreme opposite of Block 1. The lack of memories and thus of additional restrictions in cell placement and routing allows higher *fanout* constraints without incurring in a clock tree structure degradation.

On the analysed block, the optimal *fanout* chosen was 128. Although at a *fanout* constraint of 256 provided slightly better power improvement, the increase of *skew* and *insertion delay* over the *fanout* constraint of 128 made it so it was discarded.

6.3. Clock Cell Analysis

The usage and selection of clock cells is highly determined by the physical layout of a block as it will set the maximum achievable *fanout* constraint without incurring in degradation of the clock structure.

This degradation appears due to the clock building algorithm as high *fanout* nets are weighted against low *fanout* nets as it results in an overall decrease of the number of clock cells that must be used.

However, this increase on the maximum *fanout* can incur in power increases due to increased wirelength due to separation between clock cells and flip-flops.

Moreover, due to the application of *Concurrent Clock and Data Optimization* and the correlation between datapath and clock structure to minimize the *Worst Negative Slack*, these penalizations in terms of power and timing can increase providing no benefit at all due to the reduction in clock cells.

Having this into consideration, two main experiments were performed regarding clock cells, the usage of a bigger set of *repeaters* and a smaller set of *repeaters* at each of the experiment blocks.

The use of bigger *repeaters* is conditioned by the downstream capacitance, and thus maximum *fanout* expected at a given clock cell. If it is possible to achieve high *fanouts* on a given design, increasing the clock cell and thus its driving power will result on having a much higher number of nets having a higher *fanout* constraint.

6.3.1. Block 1

On Block 1, the usage of bigger *repeaters* was not useful due to the limited *fanout*, resulting on a degradation of the *Quality of Results* with *fanout* constraints higher than 32.

On the other hand, the usage of smaller *repeaters*, results on a degradation of the clock structure, in this case due to clock branches capacitance balance and the application of *Concurrent Clock and Data Optimization*.

The optimal result on this case is the usage of the reference clock cells and a *fanout* constraint of 32.

6.3.2. Block 2

The placement of memories on this case has less of a negative effect due to the physical layout of this block allowing an optimal constraint of 64.

Akin to the results obtained on Block 1, the limitation on the achievable *fanout* limits the usage of bigger clock cells.

The usage of smaller cells yields worse results due to clock tree balance.

The optimal result on this case is using the reference clock cells and a *fanout* constraint of 64.

6.3.3. Block 3

At Block 3, as it has been seen, it is a block with no memory instances. This simplifies the design and allows a much smoother clock structure with a much higher *fanout* constraint. At the *fanout* analysis, it was tested that the best results obtained were using a *fanout* constraint of 128, however at 256, there were not heavy penalizations obtained in terms of power or timing.

Due to that, when using bigger clock cells, it was observed that the design did not suffer much in terms of *Worst Negative Slack*, mainly due to good timing results. Moreover, when using a *fanout* constraint of 256, improvements were observed in *skew*, *insertion delay* and number of instances used. The power gains at the clock structure were modest as the bigger clock cells have greater power consumption.

It should be noted, that at Block 3, there are twice as many *ICG*s compared to the number of *repeater*s, 14595 *ICG*s and 7086 *repeater*s, in the reference block.

This fixes a huge part of the power consumption as *ICG*s will typically have higher power consumption and cannot be reduced at the design as they are needed for clock gating strategies. This limits in all cases the possible power gains on the clock structure.

The optimal results on this case are:

- *Fanout* constraint of 256 using the bigger clock cells with best results in *skew*, *insertion delay*, and *repeater* instances.
- *Fanout* constraint of 128 using the reference clock cells with best results in *Worst Negative Slack*, clock power and global power with improvements over the reference block in *skew*, *latency* and clock *repeater*s.

6.4. Concurrent Clock and Data Optimization Analysis

The application of *Concurrent Clock and Data Optimization* is usually recommended for high frequency circuits. This need becomes more prevalent when the physical layout contains blockages or has flip-flop placement in-between memories that limits the maximum *fanout* and degrades the *Worst Negative Slack*.

The improvements in terms of *repeater* instances, *skew*, *insertion delay* and clock power are not enough to justify the heavy degradation in timing metrics and overall power due to the lack of optimization of the datapath considering the clockpath.

Moreover, the improvements in terms of *skew* and *latency* depend on how the initial physical layout is. On Block 2, where less memory instances are used and where less flip-flops are placed in-between memories, the improvements obtained are more moderate.

Due to the results obtained, this experiment set was not performed in Block 3.

Overall, the use of *Concurrent Clock and Data Optimization* is mandatory in the blocks analysed.

6.5. Clock Placement Analysis

In all the blocks where the pin placement has been tested some general results are obtained:

- *Insertion delay* improvement due to the reduction on the maximum clockpath length.
- Increase on the overall wirelength due to *Concurrent Clock and Data Optimization* implementation.
- Increase on the number of clock cells used compared to the base placement of the clock pin.
- Increase on the *Worst Negative Slack*.

From the blocks analysed, the best results obtained in terms of *skew* are obtained on Block 3 where there are no memories on the floorplan. On the other cases, the placement of memories and thus the added limitations on clock routing limit the improvements of *skew* to be obtained.

In all blocks where a *fanout* constraint higher than 32 is applied and the clock pin has been moved, the maximum *fanout* constraint has not been used. It is likely that errors appeared during the runs performed overriding the constraint that was set.

The modification of the clock placement also results on the overall degradation of the *Worst Negative Slack* and *Total Negative Slack*. By modifying the clock pin input, it is possible to unbalance the different clock paths asides from possible problems with memory placements amongst other things.

Overall, in all the blocks, the movement of the clock pin has yielded worse results than the ones obtained in the reference block in terms of *Worst Negative Slack*, *repeater* instances and power.

In order to improve the possible results obtained on this case several conditions should be taken into consideration:

- Study of flip-flop density and memory placement to place the clock pin.
- Placement of the clock pin input in the top-design block layers.

6.6. Future development

During the development of the project several parts could have been modified. The initial experiment set could have been performed correctly in order to obtain relevant data with the refined methodology used on the rest of the experiment sets.

Several results obtained also yielded errors limiting the usefulness of certain parts of the analysis, which should be redone.

Parts of the original thesis were left out such as the analysis of multi-source clock structures and further analysis using *PrimeTime*.

It should also be tested if the individual optimal results obtained in the experiment sets can be used together without incurring in metrics degradation.

Bibliography

[1] *Solvnet Synopsys Online Help* [Online] Available: https://www.solvnet.synopsys.com

[2] S. Tam. *Clocking in Modern VLSI Systems,* Springer-Verlag, USA, 2009.

[3] J. Kim, D. Joo, T. Kim. "An optimal algorithm of adjustable delay buffer insertion for solving clock *skew* variation problem" *In 2013 50th ACM/EDAC/IEE Design Automation Conference, 2013,* 29 May -7 June, Austin, USA. pp. 1-6, July 2013.

[4] R. Ewetz, S. Janarthanan, C. Koh. "Construction of reconfigurable clock trees for MCMM designs" *In Design Automation Conference (DAC), 2015 52nd ACM/EDAC/IEEE, 2015,* 8-12 June 2015, San Francisco, USA. pp. 1-6. doi: 10.1145/2744769.2744811.

[5] C.Deng, Y. CAi, Q. Zhou, Z. Chen "An efficient buffer sizing algorithm for clock trees considering process variations" *In 6th Asia Symposium on Quality Electronic Design, ASQED 2015,* 4-5 August 2015 Kula Lumpur, Malaysia. pp. 108-113 doi: 10.1109/ACQED.2015.7274017.

[6] J. Kawa. "*FinFET* Design, Manufacturability, and Reliability" Synopsys. [Online] Available: https://www.synopsys.com/designware-ip/technical-bulletin/*FinFET*-design.html [Access: 3 October 2017]

[7] M. Hartman, J. Taylor. "Design How-To Solve leakage and dynamic power loss" [Online] Available: https://www.eetimes.com/document.asp?doc_id=1275132 [Access: 1 October 2017]

[8] I. Ringworm. "Double Gate *FinFET*" [Online] Available: https://commons.wikimedia.org/wiki/File:Doublegate_*FinFET*.PNG [Access: 10 October 2017]

Appendices

This section will cover all the additional data used during the analysis. It will be provided the raw data and comparisons upon it has been done. All the user-made scripts will also be added.

Block 1 results:

Slew Clock QoR metrics											
Set	WNS (ns)	TNS (ns)	Skew (ns)	Latency (ns)	Clock Cells	Clock Repeaters	Area Clock Cells (um2)	Repeater Area (um2)	Dynamic Clock Power (uW)	Leakage Clock Power (uW)	Total Clock Power (uW)
reference_run	-0.346	-581.8	0.598	1.37	33161	28969	8771.69	7095.875	722639.71	1329.525	723969.235
slew100	-0.497	-746.1	0.609	1.325	34909	30718	9176.871	7501.141	746510.595	1400.738	747911.333
slew60	-0.441	-809.3	0.587	1.207	35235	31044	9142.199	7466.469	746462.848	1392.468	747855.316
slew40	-0.509	-925	0.649	1.4	41789	37598	10582.308	8906.579	766330.606	1617.621	767948.227
Slew General QoR metrics											
Set	Utilization	Total Stdcell Area (um2)	Total Dynamic Power (uW)	Total Leakage Power (uW)	Total Power (uW)	Number of DRCs					
reference_run	0.3186	792194.1384	1.17E+06	2.68E+05	1.44E+06	62607					
slew100	0.3121	776143.4184	1.20E+06	3.12E+05	1.51E+06	52118					
slew60	0.3108	772940.6429	1.20E+06	3.15E+05	1.52E+06	87510					
slew40	0.3166	787205.706	1.22E+06	3.21E+05	1.54E+06	76238					
Clock QoR Comparison											
Set	WNS (%)	TNS (%)	Skew (%)	Latency (%)	Clock Cells (%)	Clock Repeaters (%)	Area Clock Cells (%)	Repeater Area (%)	Dynamic Clock Power (%)	Leakage Clock Power (%)	Total Clock Power (%)
slew100	43.6416185	28.239945	1.839464883	-3.284671533	5.271252375	6.03748835	4.619189689	5.71129001	3.303289962	5.356273857	3.30706014
slew60	27.4566474	39.10278446	-1.839464883	-11.89781022	6.254334911	7.162829231	4.223918082	5.222668099	3.296682658	4.734247194	3.299322657
slew40	47.10982659	58.98934342	8.528428094	2.189781022	26.01851573	29.7870137	20.64160954	25.51769866	6.046013718	21.66909235	6.074704542
General QoR Comparison											
Set	Utilization (%)	Total Stdcell Area (%)	Total Dynamic Power (%)	Total Leakage Power (%)	Total Power (%)	Number of DRCs (%)					
slew100	-2.040175769	-2.026109412	2.561912895	16.18195377	5.072967338	-16.75371764					
slew60	-2.448210923	-2.430401156	2.647309991	17.59880686	5.489923558	39.77670229					
slew40	-0.62774639	-0.629698221	4.269854825	19.79865772	7.157748436	21.77232578					

Appendix 1: *Slew* Data Comparison for Block 1.

Bigger Repeaters Clock QoR metrics											
Set	WNS (ns)	TNS (ns)	Skew (ns)	Latency (ns)	Clock Cells	Clock Repeaters	Area Clock Cells (um2)	Repeater Area (um2)	Dynamic Clock Power (uW)	Leakage Clock Power (uW)	Total Clock Power (uW)
reference_run	-0.346	-581.8	0.598	1.37	33161	28969	8771.69	7095.875	722639.71	1329.525	723969.235
fanout256	-0.566	-990.6	0.506	1.248	28538	24347	7608.784	5933.054	736840.221	1129.969	737970.19
ccd_32_24-16-8	-0.662	-1302.9	0.707	1.364	37663	33472	12546.551	10870.821	802971.812	2065.426	805037.238
ccd_256_24-16-8	-0.634	-836	0.681	1.36	31840	27649	10878.732	9203.003	790426.755	1743.659	792170.414
Bigger Repeaters General QoR metrics											
Set	Utilization	Total Stdcell Area (um2)	Total Dynamic Power (uW)	Total Leakage Power (uW)	Total Power (uW)	Number of DRCs					
reference_run	0.3186	792194.1384	1.17E+06	2.68E+05	1.44E+06	62607					
fanout256	0.3104	771863.3026	1.19E+06	3.10E+05	1.50E+06	67696					
ccd_32_24-16-8	0.3167	787506.2714	1.26E+06	3.19E+05	1.58E+06	70852					
ccd_256_24-16-8	0.3195	794432.6426	1.25E+06	3.14E+05	1.56E+06	98690					
Clock QoR Comparison											
Set	WNS (%)	TNS (%)	Skew (%)	Latency (%)	Clock Cells (%)	Clock Repeaters (%)	Area Clock Cells (%)	Repeater Area (%)	Dynamic Clock Power (%)	Leakage Clock Power (%)	Total Clock Power (%)
fanout256	63.58381503	70.26469577	-15.38461538	-8.905109489	-13.94107536	-15.95498636	-13.25749086	-16.38728134	1.965088661	-15.00957109	1.933915742
ccd_32_24-16-8	91.32947977	123.9429357	18.22742475	-0.437956204	13.5761889	15.54420242	43.03459197	53.19916148	11.11648044	55.35067035	11.19771381
ccd_256_24-16-8	83.23699422	43.69199037	13.87959866	-0.729927007	-3.983595187	-4.556594981	24.02093553	29.6951116	9.38047606	31.14901939	9.420452652
General QoR Comparison											
Set	Utilization (%)	Total Stdcell Area (%)	Total Dynamic Power (%)	Total Leakage Power (%)	Total Power (%)	Number of DRCs (%)					
fanout256	-2.573760201	-2.566395636	1.964133219	15.398956	4.517025712	8.128484035					
ccd_32_24-16-8	-0.596359071	-0.591757345	7.429547395	18.7546607	9.589993051	13.16945389					
ccd_256_24-16-8	0.282485876	0.282570154	6.746370623	17.22595078	8.686587908	57.63413037					

Appendix 2: *Fanout* Data Comparison for Block 1.

Bigger Repeaters Clock QoR metrics											
Set	WNS (ns)	TNS (ns)	Skew (ns)	Latency (ns)	Clock Cells	Clock Repeaters	Area Clock Cells (um2)	Repeater Area (um2)	Dynamic Clock Power (uW)	Leakage Clock Power (uW)	Total Clock Power (uW)
reference_run	-0.346	-581.8	0.598	1.37	33161	28969	8771.69	7095.875	722639.71	1329.525	723969.235
fanout256	-0.566	-990.6	0.506	1.248	28538	24347	7608.784	5933.054	736840.221	1129.969	737970.19
ccd_32_24-16-8	-0.662	-1302.9	0.707	1.364	37663	33472	12546.551	10870.821	802971.812	2065.426	805037.238
ccd_256_24-16-8	-0.634	-836	0.681	1.36	31840	27649	10878.732	9203.003	790426.755	1743.659	792170.414
Bigger Repeaters General QoR metrics											
Set	Utilization	Total Stdcell Area (um2)	Total Dynamic Power (uW)	Total Leakage Power (uW)	Total Power (uW)	Number of DRCs					
reference_run	0.3186	792194.1384	1.17E+06	2.68E+05	1.44E+06	62607					
fanout256	0.3104	771863.3026	1.19E+06	3.10E+05	1.50E+06	67696					
ccd_32_24-16-8	0.3167	787506.2714	1.26E+06	3.19E+05	1.58E+06	70852					
ccd_256_24-16-8	0.3195	794432.6426	1.25E+06	3.14E+05	1.56E+06	98690					
Clock QoR Comparison											
Set	WNS (%)	TNS (%)	Skew (%)	Latency (%)	Clock Cells (%)	Clock Repeaters (%)	Area Clock Cells (%)	Repeater Area (%)	Dynamic Clock Power (%)	Leakage Clock Power (%)	Total Clock Power (%)
fanout256	63.58381503	70.26469577	-15.38461538	-8.905109489	-13.94107536	-15.95498636	-13.25749086	-16.38728134	1.965088661	-15.00957109	1.933915742
ccd_32_24-16-8	91.32947977	123.9429357	18.22742475	-0.437956204	13.5761889	15.54420242	43.03459197	53.19916148	11.11648044	55.35067035	11.19771381
ccd_256_24-16-8	83.23699422	43.69199037	13.87959866	-0.729927007	-3.983595187	-4.556594981	24.02093553	29.6951116	9.38047606	31.14901939	9.420452652
General QoR Comparison											
Set	Utilization (%)	Total Stdcell Area (%)	Total Dynamic Power (%)	Total Leakage Power (%)	Total Power (%)	Number of DRCs (%)					
fanout256	-2.573760201	-2.566395636	1.964133219	15.398956	4.517025712	8.128484035					
ccd_32_24-16-8	-0.596359071	-0.591757345	7.429547395	18.7546607	9.589993051	13.16945389					
ccd_256_24-16-8	0.282485876	0.282570154	6.746370623	17.22595078	8.686587908	57.63413037					

Appendix 3: Bigger *Repeater* Set Data Comparison for Block 1.

Smaller Repeaters Clock QoR metrics											
Set	WNS (ns)	TNS (ns)	Skew (ns)	Latency (ns)	Clock Cells	Clock Repeaters	Area Clock Cells (um2)	Repeater Area (um2)	Dynamic Clock Power (uW)	Leakage Clock Power (uW)	Total Clock Power (uW)
reference_run	-0.346	-581.8	0.598	1.37	33161	28969	8771.69	7095.875	722639.71	1329.525	723969.235
fanout64	-0.56	-872.2	0.587	1.314	34743	30552	8962.985	7287.255	748887.924	1360.174	750248.097
ccd32_14104	-0.577	-853.1	0.545	1.298	28480	24289	6366.909	4691.18	694170.666	883.809	695054.475
ccd64_14104	-0.529	-848.5	0.594	1.306	35585	31394	9173.501	7497.771	754713.571	1395.08	756108.651
Smaller Repeaters General QoR metrics											
Set	Utilization	Total Stdcell Area (um2)	Total Dynamic Power (uW)	Total Leakage Power (uW)	Total Power (uW)	Number of DRCs					
reference_run	0.3186	792194.1384	1.17E+06	2.68E+05	1.44E+06	62607					
fanout64	0.3134	779322.6319	1.21E+06	3.18E+05	1.53E+06	91461					
ccd32_14104	0.3141	780970.0013	1.15E+06	3.12E+05	1.46E+06	51076					
ccd64_14104	0.3079	765687.4596	1.21E+06	3.09E+05	1.52E+06	104297					
Clock QoR Comparison											
Set	WNS (%)	TNS (%)	Skew (%)	Latency (%)	Clock Cells (%)	Clock Repeaters (%)	Area Clock Cells (%)	Repeater Area (%)	Dynamic Clock Power (%)	Leakage Clock Power (%)	Total Clock Power (%)
fanout64	61.84971098	49.91405981	-1.839464883	-4.087591241	4.770664335	5.464462011	2.180822624	2.697059912	3.632268423	2.305259397	3.629831315
ccd32_14104	66.76300578	46.63114472	-8.862876254	-5.255474453	-14.11597961	-16.15520039	-27.41525293	-33.88863248	-3.939590311	-33.52445422	-3.993921095
ccd64_14104	52.89017341	45.84049502	-0.668896321	-4.671532847	7.309791623	8.371017294	4.580770638	5.663797629	4.438430459	4.930708336	4.439334497
General QoR Comparison											
Set	Utilization (%)	Total Stdcell Area (%)	Total Dynamic Power (%)	Total Leakage Power (%)	Total Power (%)	Number of DRCs (%)					
fanout64	-1.632140615	-1.624791939	3.15969257	18.68008949	6.045865184	46.0874982					
ccd32_14104	-1.412429379	-1.416841726	-2.220324509	16.29381059	1.250868659	-18.41806827					
ccd64_14104	-3.358443189	-3.345982697	3.15969257	15.21252796	5.420430855	66.58999792					

Appendix 4: Smaller *Repeater* Set Data Comparison for Block 1.

Smaller Repeaters Clock QoR metrics											
Set	WNS (ns)	TNS (ns)	Skew (ns)	Latency (ns)	Clock Cells	Clock Repeaters	Area Clock Cells (um2)	Repeater Area (um2)	Dynamic Clock Power (uW)	Leakage Clock Power (uW)	Total Clock Power (uW)
reference_run	-0.346	-581.8	0.598	1.37	33161	28969	8771.69	7095.875	722639.71	1329.525	723969.235
noccd_route_local	-0.569	-1400.3	0.135	1.049	29513	25322	8085.022	6409.292	713926.249	1236.565	715162.814
noccd_custom_local	-0.787	-1583.7	0.577	1.359	28330	24139	7980.721	6304.991	714527.096	1222.844	715749.939
noccd_route_nolocal	-0.713	-1810	0.548	1.354	29967	25776	7771.29	6095.561	713722.291	1145.161	714867.451
Smaller Repeaters General QoR metrics											
Set	Utilization	Total Stdcell Area (um2)	Total Dynamic Power (uW)	Total Leakage Power (uW)	Total Power (uW)	Number of DRCs					
reference_run	0.3186	792194.1384	1.17E+06	2.68E+05	1.44E+06	62607					
noccd_route_local	0.3057	760266.1147	1.17E+06	3.24E+05	1.49E+06	58041					
noccd_custom_local	0.3128	777911.2538	1.17E+06	3.29E+05	1.50E+06	75924					
noccd_route_nolocal	0.3097	770183.8889	1.16E+06	3.32E+05	1.50E+06	62541					
Clock QoR Comparison											
Set	WNS (%)	TNS (%)	Skew (%)	Latency (%)	Clock Cells (%)	Clock Repeaters (%)	Area Clock Cells (%)	Repeater Area (%)	Dynamic Clock Power (%)	Leakage Clock Power (%)	Total Clock Power (%)
noccd_route_local	64.45086705	140.6840839	-77.42474916	-23.43065693	-11.00087452	-12.58931962	-7.828229224	-9.675804605	-1.205782201	-6.991970817	-1.216408181
noccd_custom_local	127.4566474	172.206944	-3.511705686	-0.802919708	-14.56831821	-16.67299527	-9.017293133	-11.14568675	-1.122636064	-8.023993532	-1.135310121
noccd_route_nolocal	106.0693642	211.103472	-8.361204013	-1.167883212	-9.631796387	-11.0221271	-11.40487181	-14.09711981	-1.234006224	-13.86690735	-1.257205909
General QoR Comparison											
Set	Utilization (%)	Total Stdcell Area (%)	Total Dynamic Power (%)	Total Leakage Power (%)	Total Power (%)	Number of DRCs (%)					
noccd_route_local	-4.048964218	-4.030328193	-0.426985482	20.84265474	3.544127867	-7.293114189					
noccd_custom_local	-1.820464532	-1.802952573	0.085397096	22.66964952	4.308547603	21.27078442					
noccd_route_nolocal	-2.793471438	-2.778390856	-0.597779675	23.60178971	3.961084086	-0.105419522					

Appendix 5: *CCD* Set Data Comparison for Block 1.

Moved Clock Pin Clock QoR metrics											
Set	WNS (ns)	TNS (ns)	Skew (ns)	Latency (ns)	Clock Cells	Clock Repeaters	Area Clock Cells (um2)	Repeater Area (um2)	Dynamic Clock Power (uW)	Leakage Clock Power (uW)	Total Clock Power (uW)
reference_run	-0.346	-581.8	0.598	1.37	33161	28969	8771.69	7095.875	722639.71	1329.525	723969.235
reference_moved	-0.391	-774	0.593	1.079	34874	30683	8988.318	7312.588	749885.134	1365.216	751250.35
Moved Clock Pin General QoR metrics											
Set	Utilization	Total Stdcell Area (um2)	Total Dynamic Power (uW)	Total Leakage Power (uW)	Total Power (uW)	Number of DRCs					
reference_run	0.3186	792194.1384	1.17E+06	2.68E+05	1.44E+06	62607					
reference_moved	0.3047	757709.7662	1.20E+06	3.07E+05	1.50E+06	93314					
Clock QoR Comparison											
Set	WNS (%)	TNS (%)	Skew (%)	Latency (%)	Clock Cells (%)	Clock Repeaters (%)	Area Clock Cells (%)	Repeater Area (%)	Dynamic Clock Power (%)	Leakage Clock Power (%)	Total Clock Power (%)
reference_moved	13.00578035	33.03540736	-0.836120401	-21.24087591	5.165706704	5.916669543	2.46962672	3.054070146	3.770263884	2.684492582	3.768269932
General QoR Comparison											
Set	Utilization (%)	Total Stdcell Area (%)	Total Dynamic Power (%)	Total Leakage Power (%)	Total Power (%)	Number of DRCs (%)					
reference_moved	-4.362837414	-4.353020368	2.134927412	14.35495899	4.447533009	49.04723114					

Appendix 6: Moved Clock Pin Data Comparison for Block 1.

Slew Clock QoR metrics											
Set	WNS (ns)	TNS (ns)	Skew (ns)	Latency (ns)	Clock Cells	Clock Repeaters	Area Clock Cells (um2)	Repeater Area (um2)	Dynamic Clock Power (uW)	Leakage Clock Power (uW)	Total Clock Power (uW)
reference_run	-0.06	-1.6	0.206	0.762	27032	18036	6396.44	4211.829	429350.789	962.957	430313.746
slew100	-0.134	-1.7	0.169	0.773	27731	18733	6576.683	4391.9	436811.415	992.236	437803.651
slew60	-0.186	-2.5	0.188	0.77	29557	20560	7138.401	4953.703	433174.586	1091.385	434265.971
slew40	-0.152	-1.1	0.214	0.799	41292	32295	9840.319	7655.622	460323.178	1533.174	461856.352
Slew General QoR metrics											
Set	Utilization	Total Stdcell Area (um2)	Total Dynamic Power (uW)	Total Leakage Power (uW)	Total Power (uW)	Number of DRCs					
reference_run	0.4576	570155.2752	8.75E+05	6.72E+04	9.42E+05	13289					
slew100	0.457	569390.1814	8.88E+05	6.72E+04	9.55E+05	11833					
slew60	0.4545	566276.5702	8.82E+05	6.52E+04	9.47E+05	12492					
slew40	0.4617	575355.5657	9.07E+05	6.92E+04	9.76E+05	13094					
Clock QoR Comparison											
Set	WNS (%)	TNS (%)	Skew (%)	Latency (%)	Clock Cells (%)	Clock Repeaters (%)	Area Clock Cells (%)	Repeater Area (%)	Dynamic Clock Power (%)	Leakage Clock Power (%)	Total Clock Power (%)
slew100	123.33333	6.25	-17.96116505	1.443569554	2.585824208	3.864493236	2.817864312	4.275363506	1.737652798	3.040530366	1.740568381
slew60	210	56.25	-8.737864078	1.049868766	9.340781296	13.99423375	11.5995929	17.61405793	0.890599738	13.33683643	0.918451952
slew40	153.33333	-31.25	3.883495146	4.855643045	52.75229358	79.05854957	53.84055819	81.76478675	7.213772466	59.21520899	7.330141389
General QoR Comparison											
Set	Utilization (	Total Stdcell Area (%)	Total Dynamic Power (%)	Total Leakage Power (%)	Total Power (%)	Number of DRCs (%)					
slew100	-0.1311189	-0.134190427	1.474454223	0	1.369426752	-10.95643013					
slew60	-0.6774476	-0.680289242	0.765801806	-2.873734366	0.50955414	-5.997441493					
slew40	0.895979	0.912083204	3.646130986	3.022632519	3.609341826	-1.467379035					

Appendix 7: *Slew* Data Comparison for Block 2.

Block 2 results:

Fanout Clock QoR metrics											
Set	WNS (ns)	TNS (ns)	Skew (ns)	Latency (ns)	Clock Cells	Clock Repeaters	Area Clock Cells (um2)	Repeater Area (um2)	Dynamic Clock Power (uW)	Leakage Clock Power (uW)	Total Clock Power (uW)
reference_run	-0.06	-1.6	0.206	0.762	27032	18036	6396.44	4211.829	429350.789	962.957	430313.746
fanout64	-0.07	-3	0.272	0.736	21844	12847	5326.668	3141.971	414481.485	777.867	415259.352
fanout128	-0.123	-3.9	0.253	0.774	21791	12794	5467.098	3282.401	416227.742	809.724	417037.466
fanout256	-0.096	-3.7	0.22	0.82	24439	15442	6090.306	3905.609	433184.419	910.349	434094.768
fanout512	-0.165	-2.8	0.217	0.784	24386	15388	6165.79	3981.007	427895.235	928.163	428823.399
Fanout General QoR metrics											
Set	Utilization	Total Stdcell Area (um2)	Total Dynamic Power (uW)	Total Leakage Power (uW)	Total Power (uW)	Number of DRCs					
reference_run	0.4576	570155.2752	8.75E+05	6.72E+04	9.42E+05	13289					
fanout64	0.4549	566832.2906	8.61E+05	6.27E+04	9.24E+05	12861					
fanout128	0.4587	571506.2203	8.61E+05	6.79E+04	9.28E+05	13098					
fanout256	0.4576	570219.1925	8.78E+05	6.62E+04	9.44E+05	13043					
fanout512	0.458	570738.8131	8.77E+05	6.66E+04	9.44E+05	16781					
Clock QoR Comparison											
Set	WNS (%)	TNS (%)	Skew (%)	Latency (%)	Clock Cells (%)	Clock Repeaters (%)	Area Clock Cells (%)	Repeater Area (%)	Dynamic Clock Power (%)	Leakage Clock Power (%)	Total Clock Power (%)
fanout64	16.66666667	87.5	32.03883495	-3.412073491	-19.19206866	-28.7702373	-16.7244905	-25.40126866	-3.463206399	-19.22100364	-3.498469231
fanout128	105	143.75	22.81553398	1.57480315	-19.38813258	-29.06409403	-14.52905053	-22.06708772	-3.056486057	-15.91275623	-3.085255845
fanout256	60	131.25	6.796116505	7.611548556	-9.59233501	-14.38234642	-4.786005966	-7.270475606	0.892889939	-5.463172291	0.87866633
fanout512	175	75	5.339805825	2.887139108	-9.788398935	-14.68174762	-3.605912039	-5.480326955	-0.339012769	-3.613245451	-0.346339622
General QoR Comparison											
Set	Utilization (%)	Total Stdcell Area (%)	Total Dynamic Power (%)	Total Leakage Power (%)	Total Power (%)	Number of DRCs (%)					
fanout64	-0.590034965	-0.582820987	-1.600182878	-6.611078023	-1.942675159	-3.220708857					
fanout128	0.240384615	0.236943366	-1.645902389	1.057176891	-1.44373673	-1.437278953					
fanout256	0	0.011210508	0.377185964	-1.488981537	0.25477707	-1.851155091					
fanout512	0.087412587	0.10234719	0.262887187	-0.818939845	0.191082803	26.27737226					

Appendix 8: *Fanout* Data Comparison for Block 2.

Bigger Repeaters Clock QoR metrics											
Set	WNS (ns)	TNS (ns)	Skew (ns)	Latency (ns)	Clock Cells	Clock Repeaters	Area Clock Cells (um2)	Repeater Area (um2)	Dynamic Clock Power (uW)	Leakage Clock Power (uW)	Total Clock Power (uW)
reference_run	-0.06	-1.6	0.206	0.762	27032	18036	6396.44	4211.829	429350.789	962.957	430313.746
fanout256	-0.096	-3.7	0.22	0.82	24439	15442	6090.306	3905.609	433184.419	910.349	434094.768
ccd_32_24-16-8	-0.138	-2.6	0.208	0.784	26633	17631	7748.385	5563.26	442784.367	1250.884	444035.251
ccd_256_24-16-8	-0.125	-9.7	0.283	0.873	22722	13724	6826.84	4642.057	434365.334	1072.2	435437.534
Bigger Repeaters General QoR metrics											
Set	Utilization	Total Stdcell Area (um2)	Total Dynamic Power (uW)	Total Leakage Power (uW)	Total Power (uW)	Number of DRCs					
reference_run	0.4576	570155.2752	8.75E+05	6.72E+04	9.42E+05	13289					
fanout256	0.4576	570219.1925	8.78E+05	6.62E+04	9.44E+05	13043					
ccd_32_24-16-8	0.4576	570244.5538	8.88E+05	6.62E+04	9.54E+05	11644					
ccd_256_24-16-8	0.4632	577104.8371	8.83E+05	7.19E+04	9.55E+05	10596					
Clock QoR Comparison											
Set	WNS (%)	TNS (%)	Skew (%)	Latency (%)	Clock Cells (%)	Clock Repeaters (%)	Area Clock Cells (%)	Repeater Area (%)	Dynamic Clock Power (%)	Leakage Clock Power (%)	Total Clock Power (%)
fanout256	60	131.25	6.796116505	7.611548556	-9.59233501	-14.38234642	-4.786005966	-7.270475606	0.892889939	-5.463172291	0.87866633
ccd_32_24-16-8	130	62.5	0.970873786	2.887139108	-1.476028411	-2.245508982	21.13589747	32.08655907	3.128811765	29.90029669	3.18872105
ccd_256_24-16-8	108.3333333	506.25	37.37864078	14.56692913	-15.94406629	-23.90774008	6.728742863	10.21475468	1.167936598	11.34453563	1.190709813
General QoR Comparison											
Set	Utilization (%)	Total Stdcell Area (%)	Total Dynamic Power (%)	Total Leakage Power (%)	Total Power (%)	Number of DRCs (%)					
fanout256	0	0.011210508	0.377185964	-1.488981537	0.25477707	-1.851155091					
ccd_32_24-16-8	0	0.015658647	1.463024346	-1.444312091	1.263269639	-12.37865904					
ccd_256_24-16-8	1.223776224	1.218889345	0.880100583	7.057772484	1.326963907	-20.26488073					

Appendix 9: Bigger *Repeater* Set Data Comparison for Block 2.

Smaller Repeaters Clock QoR metrics											
Set	WNS (ns)	TNS (ns)	Skew (ns)	Latency (ns)	Clock Cells	Clock Repeaters	Area Clock Cells (um2)	Repeater Area (um2)	Dynamic Clock Power (uW)	Leakage Clock Power (uW)	Total Clock Power (uW)
reference_run	-0.06	-1.6	0.206	0.762	27032	18036	6396.44	4211.829	429350.789	962.957	430313.746
fanout64	-0.07	-3	0.272	0.736	21844	12847	5326.668	3141.971	414481.485	777.867	415259.352
ccd32_14104	-0.092	-3	0.204	0.729	26540	17543	5640.886	3456.188	420373.182	799.444	421172.626
ccd64_14104	-0.131	-1.1	0.211	0.773	23427	14429	5395.47	3210.687	414770.937	788.01	415558.947
Smaller Repeaters General QoR metrics											
Set	Utilization	Total Stdcell Area (um2)	Total Dynamic Power (uW)	Total Leakage Power (uW)	Total Power (uW)	Number of DRCs					
reference_run	0.4576	570155.2752	8.75E+05	6.72E+04	9.42E+05	13289					
fanout64	0.4549	566832.2906	8.61E+05	6.27E+04	9.24E+05	12861					
ccd32_14104	0.4562	568389.9816	8.63E+05	6.83E+04	9.31E+05	103953					
ccd64_14104	0.4579	570519.1582	8.63E+05	6.66E+04	9.30E+05	10267					
Clock QoR Comparison											
Set	WNS (%)	TNS (%)	Skew (%)	Latency (%)	Clock Cells (%)	Clock Repeaters (%)	Area Clock Cells (%)	Repeater Area (%)	Dynamic Clock Power (%)	Leakage Clock Power (%)	Total Clock Power (%)
fanout64	16.66666667	87.5	32.03883495	-3.412073491	-19.19206866	-28.7702373	-16.7244905	-25.40126866	-3.463206399	-19.22100364	-3.498469231
ccd32_14104	53.33333333	87.5	-0.970873786	-4.330708661	-1.820065108	-2.733422045	-11.81210173	-17.94092305	-2.090972517	-16.9803013	-2.124291888
ccd64_14104	118.3333333	-31.25	2.427184466	1.443569554	-13.33604617	-19.99889111	-15.64886093	-23.76976843	-3.395790196	-18.16768558	-3.428846775
General QoR Comparison											
Set	Utilization (%)	Total Stdcell Area (%)	Total Dynamic Power (%)	Total Leakage Power (%)	Total Power (%)	Number of DRCs (%)					
fanout64	-0.590034965	-0.582820987	-1.6	-6.696428571	-1.910828025	-3.220708857					
ccd32_14104	-0.305944056	-0.309616288	-1.371428571	1.636904762	-1.167728238	682.2484762					
ccd64_14104	0.065559441	0.063821737	-1.371428571	-0.892857143	-1.27388535	-22.74061254					

Appendix 10: Smaller *Repeater* Set Data Comparison for Block 2.

Smaller Repeaters Clock QoR metrics												
Set	WNS (ns)	TNS (ns)	Skew (ns)	Latency (ns)	Clock Cells	Clock Repeaters	Area Clock Cells (um2)	Repeater Area (um2)	Dynamic Clock Power (uW)	Leakage Clock Power (uW)	Total Clock Power (uW)	
reference_run	-0.06	-1.6	0.206	0.762	27032	18036	6396.44	4211.829	429350.789	962.957	430313.746	
noccd_route_local	-0.312	-20.1	0.122	0.809	28432	19430	7097.331	4912.206	437564.51	1097.615	438662.125	
noccd_custom_local	-0.353	-17.3	0.124	0.784	26021	17025	6523.704	4339.092	430351.475	999.851	431351.326	
noccd_route_nolocal	-0.311	-25.1	0.142	0.76	25455	16458	6030.644	3845.947	423600.927	897.865	424498.792	
Smaller Repeaters General QoR metrics												
Set	Utilization	Total Stdcell Area (um2)	Total Dynamic Power (uW)	Total Leakage Power (uW)	Total Power (uW)	Number of DRCs						
reference_run	0.4576	570155.2752	8.75E+05	6.72E+04	9.42E+05	13289						
noccd_route_local	0.4614	574917.5981	8.83E+05	7.81E+04	9.61E+05	13705						
noccd_custom_local	0.4585	571336.0884	8.73E+05	7.54E+04	9.49E+05	12485						
noccd_route_nolocal	0.4594	572472.6622	8.70E+05	7.77E+04	9.47E+05	11104						
Clock QoR Comparison												
Set	WNS (%)	TNS (%)	Skew (%)	Latency (%)	Clock Cells (%)	Clock Repeaters (%)	Area Clock Cells (%)	Repeater Area (%)	Dynamic Clock Power (%)	Leakage Clock Power (%)	Total Clock Power (%)	
noccd_route_local	420	1156.25	-40.77669903	6.167979003	5.179047055	7.728986472	10.95751699	16.628809	1.913055993	13.98380198	1.940067934	
noccd_custom_local	488.3333333	981.25	-39.80582524	2.887139108	-3.740011838	-5.605455755	1.989606719	3.021561417	0.233069561	3.831323725	0.241121742	
noccd_route_nolocal	418.3333333	1468.75	-31.06796117	-0.262467192	-5.833826576	-8.74916833	-5.718743551	-8.687009848	-1.339199123	-6.759595704	-1.351328898	
General QoR Comparison												
Set	Utilization (%)	Total Stdcell Area (%)	Total Dynamic Power (%)	Total Leakage Power (%)	Total Power (%)	Number of DRCs (%)						
noccd_route_local	0.83041958	0.835267708	0.925820094	16.22989875	2.027600849	3.130408609						
noccd_custom_local	0.196678322	0.207103793	-0.182878043	12.2096486	0.711252654	-6.050116638						
noccd_route_nolocal	0.393356643	0.406448401	-0.605783518	15.6938654	0.552016985	-16.44217022						

Appendix 11: *CCD* Set Data Comparison for Block 2.

Moved Clock Pin Clock QoR metrics											
Set	WNS (ns)	TNS (ns)	Skew (ns)	Latency (ns)	Clock Cells	Clock Repeaters	Area Clock Cells (um2)	Repeater Area (um2)	Dynamic Clock Power (uW)	Leakage Clock Power (uW)	Total Clock Power (uW)
reference_run	-0.06	-1.6	0.206	0.762	27032	18036	6396.44	4211.829	429350.789	962.957	430313.746
fanout64	-0.07	-3	0.272	0.736	21844	12847	5326.668	3141.971	414481.485	777.867	415259.352
fanout64_moved	-0.121	-2.5	0.193	0.633	26825	17824	6435.796	4250.756	428838.885	972.374	429811.259
Moved Clock Pin General QoR metrics											
Set	Utilization	Total Stdcell Area (um2)	Total Dynamic Power (uW)	Total Leakage Power (uW)	Total Power (uW)	Number of DRCs					
reference_run	0.4576	570155.2752	8.75E+05	6.72E+04	9.42E+05	13289					
fanout64	0.4549	566832.2906	8.61E+05	6.27E+04	9.24E+05	12861					
fanout64_moved	0.4572	569658.5026	8.77E+05	6.50E+04	9.42E+05	14462					
Clock QoR Comparison											
Set	WNS (%)	TNS (%)	Skew (%)	Latency (%)	Clock Cells (%)	Clock Repeaters (%)	Area Clock Cells (%)	Repeater Area (%)	Dynamic Clock Power (%)	Leakage Clock Power (%)	Total Clock Power (%)
fanout64	16.66666667	87.5	32.03883495	-3.412073491	-19.19206866	-28.7702373	-16.7244905	-25.40126866	-3.463206399	-19.22100364	-3.498469231
fanout64_moved	101.6666667	56.25	-6.310679612	-16.92913386	-0.7657591	-1.175426924	0.615279749	0.924230305	-0.119227451	0.977925286	-0.11677224
General QoR Comparison											
Set	Utilization (%)	Total Stdcell Area (%)	Total Dynamic Power (%)	Total Leakage Power (%)	Total Power (%)	Number of DRCs (%)					
reference_moved	-0.590034965	-0.582820987	-1.6	-6.696428571	-1.910828025	-3.220708857					
fanout64_moved	-0.087412587	-0.087129353	0.24	-3.348214286	0	8.826849274					

Appendix 12: Moved Clock Pin Data Comparison for Block 2.

Block 3 results:

Slew Clock QoR metrics											
Set	WNS (ns)	TNS (ns)	Skew (ns)	Latency (ns)	Clock Cells	Clock Repeaters	Area Clock Cells (um2)	Repeater Area (um2)	Dynamic Clock Power (uW)	Leakage Clock Power (uW)	Total Clock Power (uW)
reference_run	-0.001	0	0.261	0.619	25164	10569	11261.779	2527.103	423489.869	1866.069	425355.938
slew100	-0.007	-1	0.145	0.534	23091	8496	10755.239	2020.563	421452.094	1783.15	423235.244
slew70	-0.003	0	0.239	0.584	26088	11493	11449.076	2714.4	422587.64	1894.401	424482.042
slew60	-0.001	0	0.217	0.543	26800	12205	11602.614	2867.938	425602.362	1918.639	427521.002
slew40	-0.003	0	0.171	0.536	29999	15404	12474.265	3739.589	433862.807	2065.309	435928.117
Slew General QoR metrics											
Set	Utilization	Total Stdcell Area (um2)	Total Dynamic Power (uW)	Total Leakage Power (uW)	Total Power (uW)	Number of DRCs					
reference_run	0.3609	460734.1469	474300	11930	486200	660					
slew100	0.3598	459311.0878	472400	11880	484300	686					
slew70	0.361	460853.5848	473200	12280	485500	717					
slew60	0.3611	460957.6574	476600	12440	489000	634					
slew40	0.3624	462575.0959	484600	12970	497600	848					
Clock QoR Comparison											
Set	WNS (%)	TNS (%)	Skew (%)	Latency (%)	Clock Cells (%)	Clock Repeaters (%)	Area Clock Cells (%)	Repeater Area (%)	Dynamic Clock Power (%)	Leakage Clock Power (%)	Total Clock Power (%)
slew100	600	0	-44.44444444	-13.73182553	-8.237958989	-19.61396537	-4.49786841	-20.04429578	-0.481186245	-4.443512003	-0.498569271
slew70	200	100	-8.429118774	-5.654281099	3.671912256	8.742548964	1.663120898	7.41153012	-0.213046183	1.518271832	-0.205450523
slew60	0	0	-16.85823755	-12.27786753	6.501351137	15.47923172	3.026475657	13.48718275	0.498829643	2.817151992	0.509000535
slew40	200	0	-34.48275862	-13.40872375	19.21395645	45.74699593	10.76638069	47.97928695	2.449394604	10.67698997	2.485489929
General QoR Comparison											
Set	Utilization (%)	Total Stdcell Area (%)	Total Dynamic Power (%)	Total Leakage Power (%)	Total Power (%)	Number of DRCs (%)					
slew100	-0.304793572	-0.30886773	-0.400590344	-0.419111484	-0.390785685	3.939393939					
slew70	0.027708507	0.025923388	-0.231920725	2.933780386	-0.143973673	8.636363636					
slew60	0.055417013	0.048511816	0.484925153	4.274937133	0.575894694	-3.939393939					
slew40	0.415627598	0.399568604	2.171621337	8.71751886	2.344714109	28.48484848					

Appendix 13: *Slew* Data Comparison for Block 3.

Fanout Clock QoR metrics											
Set	WNS (ns)	TNS (ns)	Skew (ns)	Latency (ns)	Clock Cells	Clock Repeaters	Area Clock Cells (um2)	Repeater Area (um2)	Dynamic Clock Power (uW)	Leakage Clock Power (uW)	Total Clock Power (uW)
reference_run	-0.001	0	0.261	0.619	25164	10569	11261.779	2527.103	423489.869	1866.069	425355.938
fanout64	-0.001	0	0.311	0.667	21652	7057	10471.61	1736.934	411942.47	1733.845	413676.315
fanout128	-0.002	0	0.205	0.597	21417	6822	10518.134	1783.458	411557.955	1746.139	413304.094
fanout256	-0.001	0	0.272	0.65	21113	6518	10444.363	1709.687	410351.784	1735.122	412086.906
Fanout General QoR metrics											
Set	Utilization	Total Stdcell Area (um2)	Total Dynamic Power (uW)	Total Leakage Power (uW)	Total Power (uW)	Number of DRCs					
reference_run	0.3609	460734.1469	4.74E+05	1.19E+04	4.86E+05	660					
fanout64	0.3608	460599.2009	4.63E+05	1.20E+04	4.75E+05	517					
fanout128	0.3619	462004.0387	4.63E+05	1.24E+04	4.75E+05	495					
fanout256	0.3611	460966.9966	4.61E+05	1.21E+04	4.73E+05	520					
Clock QoR Comparison											
Set	WNS (%)	TNS (%)	Skew (%)	Latency (%)	Clock Cells (%)	Clock Repeaters (%)	Area Clock Cells (%)	Repeater Area (%)	Dynamic Clock Power (%)	Leakage Clock Power (%)	Total Clock Power (%)
fanout64	0	0	19.15708812	7.754442649	-13.95644572	-33.22925537	-7.016378141	-31.26777975	-2.726723789	-7.08569726	-2.745846938
fanout128	100	0	-21.4559387	-3.554119548	-14.8903195	-35.45273914	-6.603264014	-29.42677841	-2.81752053	-6.426879178	-2.833355062
fanout256	0	0	4.214559387	5.008077544	-16.09839453	-38.3290756	-7.258320377	-32.34597086	-3.102337497	-7.017264635	-3.119512581
General QoR Comparison											
Set	Utilization (%)	Total Stdcell Area (%)	Total Dynamic Power (%)	Total Leakage Power (%)	Total Power (%)	Number of DRCs (%)					
fanout64	-0.027708507	-0.029289342	-2.320675105	0.840336134	-2.263374486	-21.66666667					
fanout128	0.277085065	0.275623547	-2.320675105	4.201680672	-2.263374486	-25					
fanout256	0.055417013	0.050538841	-2.742616034	1.680672269	-2.674897119	-21.21212121					

Appendix 14: *Fanout* Data Comparison for Block 3.

Bigger Repeaters Clock QoR metrics											
Set	WNS (ns)	TNS (ns)	Skew (ns)	Latency (ns)	Clock Cells	Clock Repeaters	Area Clock Cells (um2)	Repeater Area (um2)	Dynamic Clock Power (uW)	Leakage Clock Power (uW)	Total Clock Power (uW)
reference_run	-0.001	0	0.261	0.619	25164	10569	11261.779	2527.103	423489.869	1866.069	425355.938
fanout128	-0.002	0	0.205	0.597	21417	6822	10518.134	1783.458	411557.955	1746.139	413304.094
fanout256	-0.001	0	0.272	0.65	21113	6518	10444.363	1709.687	410351.784	1735.122	412086.906
ccd_32_24-16-8	-0.001	0	0.153	0.554	24783	10188	11871.849	3137.173	431723.836	1997.39	433721.226
ccd_128_24-16-8	-0.003	0	0.213	0.625	21681	7086	11089.562	2354.886	417989.922	1857.151	419847.073
ccd_256_24-16-8	-0.007	-1	0.142	0.533	20813	6218	10847.545	2112.869	415497.709	1815.432	417313.141
Bigger Repeaters General QoR metrics											
Set	Utilization	Total Stdcell Area (um2)	Total Dynamic Power (uW)	Total Leakage Power (uW)	Total Power (uW)	Number of DRCs					
reference_run	0.3609	460734.1469	4.74E+05	1.19E+04	4.86E+05	660					
fanout128	0.3619	462004.0387	4.63E+05	1.24E+04	4.75E+05	495					
fanout256	0.3611	460966.9966	4.61E+05	1.21E+04	4.73E+05	520					
ccd_32_24-16-8	0.3607	460416.0456	4.83E+05	1.24E+04	4.95E+05	703					
ccd_128_24-16-8	0.3636	464136.2141	4.69E+05	1.29E+04	4.82E+05	570					
ccd_256_24-16-8	0.362	462145.4107	4.66E+05	1.20E+04	4.78E+05	515					
Clock QoR Comparison											
Set	WNS (%)	TNS (%)	Skew (%)	Latency (%)	Clock Cells (%)	Clock Repeaters (%)	Area Clock Cells (%)	Repeater Area (%)	Dynamic Clock Power (%)	Leakage Clock Power (%)	Total Clock Power (%)
fanout128	100	0	-21.4559387	-3.554119548	-14.8903195	-35.45273914	-6.603264014	-29.42677841	-2.81752053	-6.426879178	-2.833355062
fanout256	0	0	4.214559387	5.008077544	-16.09839453	-38.3290756	-7.258320377	-32.34597086	-3.102337497	-7.017264635	-3.119512581
ccd_32_24-16-8	0	0	-41.37931034	-10.50080775	-1.514067716	-3.604882203	5.417172544	24.14108171	1.944312628	7.037306766	1.966655982
ccd_128_24-16-8	200	0	-18.3908046	0.969305331	-13.84120172	-32.95486801	-1.529216654	-6.814799397	-1.298719852	-0.477903014	-1.295118866
ccd_256_24-16-8	600	100	-45.59386973	-13.89337641	-17.29057384	-41.16756552	-3.678228813	-16.39165479	-1.887213977	-2.713565254	-1.890839243
General QoR Comparison											
Set	Utilization (%)	Total Stdcell Area (%)	Total Dynamic Power (%)	Total Leakage Power (%)	Total Power (%)	Number of DRCs (%)					
fanout128	0.277085065	0.275623547	-2.320675105	4.201680672	-2.263374486	-25					
fanout256	0.055417013	0.050538841	-2.742616034	1.680672269	-2.674897119	-21.21212121					
ccd_32_24-16-8	-0.055417013	-0.069042267	1.835443038	4.033613445	1.872427984	6.515151515					
ccd_128_24-16-8	0.748129676	0.738401359	-1.054852321	8.151260504	-0.843621399	-13.63636364					
ccd_256_24-16-8	0.304793572	0.30630762	-1.603375527	0.420168067	-1.58436214	-21.96969697					

Appendix 15: Bigger *Repeater* Set Data Comparison for Block 3.

Smaller Repeaters Clock QoR metrics											
Set	WNS (ns)	TNS (ns)	Skew (ns)	Latency (ns)	Clock Cells	Clock Repeaters	Area Clock Cells (um2)	Repeater Area (um2)	Dynamic Clock Power (uW)	Leakage Clock Power (uW)	Total Clock Power (uW)
reference_run	-0.001	0	0.261	0.619	25164	10569	11261.779	2527.103	423489.869	1866.069	425355.938
fanout64	-0.001	0	0.311	0.667	21652	7057	10471.61	1736.934	411942.47	1733.845	413676.315
ccd32_14104	-0.003	0	0.232	0.593	24805	10210	10698.233	1963.557	416620.506	1750.003	418370.509
ccd64_14104	-0.005	0	0.195	0.583	21935	7340	10425.742	1691.066	413623.763	1723.575	415347.338
Smaller Repeaters General QoR metrics											
Set	Utilization	Total Stdcell Area (um2)	Total Dynamic Power (uW)	Total Leakage Power (uW)	Total Power (uW)	Number of DRCs					
reference_run	0.3609	460734.1469	4.74E+05	1.19E+04	4.86E+05	660					
fanout64	0.3608	460599.2009	4.63E+05	1.20E+04	4.75E+05	517					
ccd32_14104	0.3606	460345.7023	4.68E+05	1.18E+04	4.79E+05	674					
ccd64_14104	0.3612	461106.0266	4.65E+05	1.22E+04	4.77E+05	613					
Clock QoR Comparison											
Set	WNS (%)	TNS (%)	Skew (%)	Latency (%)	Clock Cells (%)	Clock Repeaters (%)	Area Clock Cells (%)	Repeater Area (%)	Dynamic Clock Power (%)	Leakage Clock Power (%)	Total Clock Power (%)
fanout64	0	0	19.15708812	7.754442649	-13.95644572	-33.22925537	-7.016378141	-31.26777975	-2.726723789	-7.08569726	-2.745846938
ccd32_14104	200	0	-11.11111111	-4.200323102	-1.426641234	-3.396726275	-5.004058417	-22.30008037	-1.622084376	-6.219812879	-1.642254962
ccd64_14104	400	0	-25.28735632	-5.815831987	-12.83182324	-30.55161321	-7.423667255	-33.0828225	-2.329714763	-7.636052043	-2.352994071
General QoR Comparison											
Set	Utilization (%)	Total Stdcell Area (%)	Total Dynamic Power (%)	Total Leakage Power (%)	Total Power (%)	Number of DRCs (%)					
fanout64	-0.027708507	-0.029289342	-2.320675105	0.840336134	-2.263374486	-21.66666667					
ccd32_14104	-0.08312552	-0.084309922	-1.265822785	-0.840336134	-1.440329218	2.121212121					
ccd64_14104	0.08312552	0.080714595	-1.898734177	2.521008403	-1.851851852	-7.121212121					

Appendix 16: Smaller *Repeater* Set Data Comparison for Block 3.

Moved Clock Pin Clock QoR metrics											
Set	WNS (ns)	TNS (ns)	Skew (ns)	Latency (ns)	Clock Cells	Clock Repeaters	Area Clock Cells (um2)	Repeater Area (um2)	Dynamic Clock Power (uW)	Leakage Clock Power (uW)	Total Clock Power (uW)
reference_run	-0.001	0	0.261	0.619	25164	10569	11261.779	2527.103	423489.869	1866.069	425355.938
fanout128	-0.002	0	0.205	0.597	21417	6822	10518.134	1783.458	411557.955	1746.139	413304.094
reference_run_moved	-0.001	0	0.215	0.504	25431	10836	11320.841	2586.165	423633.947	1875.401	425509.348
fanout128_moved	-0.001	0	0.232	0.502	24908	10313	11181.069	2446.392	421164.578	1851.209	423015.788
fanout128_slew100_moved	-0.003	0	0.161	0.553	19257	4662	9980.349	1245.673	410150.211	1657.778	411807.989
Moved Clock Pin General QoR metrics											
Set	Utilization	Total Stdcell Area (um2)	Total Dynamic Power (uW)	Total Leakage Power (uW)	Total Power (uW)	Number of DRCs					
reference_run	0.3609	460734.1469	4.74E+05	1.19E+04	4.86E+05	660					
fanout128	0.3619	462004.0387	4.63E+05	1.24E+04	4.75E+05	495					
reference_run_moved	0.361	460814.7146	4.74E+05	1.21E+04	4.86E+05	642					
fanout128_moved	0.3605	460242.8863	4.72E+05	1.18E+04	4.84E+05	612					
fanout127_slew100_moved	0.3611	460965.7114	4.61E+05	1.21E+04	4.73E+05	526					
Clock QoR Comparison											
Set	WNS (%)	TNS (%)	Skew (%)	Latency (%)	Clock Cells (%)	Clock Repeaters (%)	Area Clock Cells (%)	Repeater Area (%)	Dynamic Clock Power (%)	Leakage Clock Power (%)	Total Clock Power (%)
fanout128	100	0	-21.4559387	-3.554119548	-14.8903195	-35.45273914	-6.603264014	-29.42677841	-2.81752053	-6.426879178	-2.833355062
reference_run_moved	0	0	-17.62452107	-18.57835218	1.06103958	2.526256032	0.524446448	2.33714257	0.034021593	0.500088689	0.036066265
fanout128_moved	0	0	-11.11111111	-18.90145396	-1.017326339	-2.422178068	-0.716671851	-3.193815211	-0.549078306	-0.796326395	-0.550162767
fanout128_slew100_moved	200	0	-38.31417625	-10.66235864	-23.47401049	-55.88986659	-11.37857527	-50.70747017	-3.149935566	-11.16202027	-3.185085193
General QoR Comparison											
Set	Utilization (%)	Total Stdcell Area (%)	Total Dynamic Power (%)	Total Leakage Power (%)	Total Power (%)	Number of DRCs (%)					
fanout128	0.277085065	0.275623547	-2.466793169	3.520536463	-2.324146442	-25					
reference_run_moved	0.027708507	0.017486809	0.021083702	1.089689858	0.041135335	-2.727272727					
fanout128_moved	-0.110834026	-0.106625611	-0.527092557	-1.089689858	-0.555327026	-7.272727273					
fanout128_slew100_moved	0.055417013	0.050259895	-2.740881299	1.257334451	-2.632661456	-20.3030303					

Appendix 17: Moved Clock Pin Data Comparison for Block 3.

Other results:

Clock Wirelength across all experiment sets:																
Block	Block 1															
Experiment Set	reference_run	slew100	slew60	slew40	fanout64	fanout128	fanout256	fanout512	ccd_32_24-16-8	ccd_256-24-16-8	ccd32_14104	ccd64_14104	noccd_local_route	noccd_local_custom	noccd_nolocal_route	reference_moved
Total Wirelength (um)	1682771.29	1815786.69	1815399.50	1886413.97	1827782.79	1672609.45	1756044.33	1723189.54	1892205.13	1857162.11	1615131.10	1854979.26	1588709.34	1581253.43	1628813.51	1856928.93
% To reference_block	100.00	107.90	107.88	112.10	108.62	99.40	104.35	102.40	112.45	110.36	95.98	110.23	94.41	93.97	96.79	110.35
Block	Block 2															
Experiment Set	reference_run	slew100	slew60	slew40	fanout64	fanout128	fanout256	fanout512	ccd_32_24-16-8	ccd_256-24-16-8	ccd32_14104	ccd64_14104	noccd_local_route	noccd_local_custom	noccd_nolocal_route	fanout64_moved
Total Wirelength (um)	1341773.04	1374205.26	1364944.85	1341692.14	1261267.90	1251015.02	1303121.56	1294262.16	1354980.17	1260463.57	1301396.39	1264796.69	1306348.96	1339627.00	1306348.96	1343597.60
% To reference_block	100.00	102.42	101.73	99.99	94.00	93.24	97.12	96.46	100.98	93.94	96.99	94.26	97.36	99.84	97.36	100.14
Block	Block 3															
Experiment Set	reference_run	slew100	slew70	slew60	slew40	fanout64	fanout128	fanout256	ccd_32_24-16-8	ccd_128_24-16-8	ccd_256-24-16-8	ccd32_14104	ccd644_14104	reference_moved	fanout128_moved	fanout128_slew100_moved
Total Wirelength (um)	981898.35	984931.32	971968.26	974390.65	989553.73	936849.36	928990.85	920691.88	987161.64	936273.78	922644.75	968134.13	934195.61	985925.12	973986.83	916847.33
% To reference_block	100.00	100.31	98.99	99.24	100.78	95.41	94.61	93.77	100.54	95.35	93.97	98.60	95.14	100.41	99.19	93.37

Appendix 18: Clock wirelength across all experiment sets and blocks.

Number of Clock Violating Paths across all experiment sets:																
Block	Block 1															
Experiment Set	reference_run	slew100	slew60	slew40	fanout64	fanout128	fanout256	fanout512	ccd_32_24-16-8	ccd_256-24-16-8	ccd32_14104	ccd64_14104	noccd_local_route	noccd_local_custom	noccd_nolocal_route	reference_moved
Number of Violating Paths	10243	11307	11417	13163	12853	10364	12621	10560	14763	12192	11630	11450	15238	13132	17585	10836
% To reference_block	100.00	110.39	111.46	128.51	125.48	101.18	123.22	103.09	144.13	119.03	113.54	111.78	148.77	128.20	171.68	105.79
Block	Block 2															
Experiment Set	reference_run	slew100	slew60	slew40	fanout64	fanout128	fanout256	fanout512	ccd_32_24-16-8	ccd_256-24-16-8	ccd32_14104	ccd64_14104	noccd_local_route	noccd_local_custom	noccd_nolocal_route	fanout64_moved
Number of Violating Paths	163	166	229	98	296	579	718	216	454	1089	278	120	1167	1040	1290	495
% To reference_block	100.00	101.84	140.49	60.12	181.60	355.21	440.49	132.52	278.53	668.10	170.55	73.62	715.95	638.04	791.41	303.68
Block	Block 3															
Experiment Set	reference_run	slew100	slew70	slew60	slew40	fanout64	fanout128	fanout256	ccd_32_24-16-8	ccd_128_24-16-8	ccd_256-24-16-8	ccd32_14104	ccd644_14104	reference_moved	fanout128_moved	fanout128_slew100_moved
Number of Violating Paths	39	532	41	16	7	32	60	41	29	91	445	37	40	27	29	39
% To reference_block	100.00	1364.10	105.13	41.03	17.95	82.05	153.85	105.13	74.36	233.33	1141.03	94.87	102.56	69.23	74.36	100.00

Appendix 19: Number of Violating Paths across all experiment sets and blocks.

Appendix 18: Script *run_gen.tcl*

```
function run_gen () {

    ### Creates the block folder, copies override, scripts and floorplan.def file
    ### Copies the floorplan.def, the scripts directory and the override directory to the generated folder.

    mkdir $1/$2
    cp -r $1/ref_run_shell/floorplan.def $1/ref_run_shell/scripts $1/ref_run_shell/override $1/$2

    ### Copies the configuration file and renames it.
    cp -r sets_folder/$3 $1/$2/override/
    mv $1/$2/override/$3 $1/$2/override/set_con.tcl

    ### Copies the reporters onto the scripts folder of the block.
    cp -r sets_folder/report_parser.tcl $1/$2/scripts
    cp -r sets_folder/report_parser_v2.tcl  $1/$2/scripts
    cp -r sets_folder/proc_QoR.tcl $1/$2/scripts
    cp -r clock_structure_analyzer.tcl $1/$2/scripts

    ### Launches a job depending on the block chosen selecting 4 cores and the needed memory with
margin.

    cd $1/$2

    echo "Following files were copied:"
    echo "report_parser.tcl generates basic report files"
    echo "proc_QoR.tcl generates timing information reports"
    echo "report_parser_v2.tcl combines reports from report_parser and proc_QoR"
    echo "clock_structure_analyzer.tcl generates a CSV file with clock structure information"

    echo "Directory was changed to $1/$2"

    if [ $1 = "Block_4" ]; then
        escad_lsf -noemail -m $mem 4-t version_tool -q $numcore -name iccjob4 "icc2_shell -f
/$1/$2/scripts/flow_run_all.tcl" -mastercpus $cpu
    fi

    if [ $1 = "Block_1" ]; then
        escad_lsf -noemail -m $mem1 -t version_tool -q $numcore -name iccjob1 "icc2_shell -f
/$1/$2/scripts/flow_run_all.tcl" -mastercpus $cpu
    fi

    if [ $1 = "Block_2" ]; then
        escad_lsf -noemail -m $mem2 -t version_tool -q $numcore -name iccjob2 "icc2_shell -f
/$1/$2/scripts/flow_run_all.tcl" -mastercpus $cpu
    fi

    if [ $1 = "Block_3" ]; then
        escad_lsf -noemail -m $mem3 -t version_tool -q $numcore -name iccjob3 "icc2_shell -f
/$1/$2/scripts/flow_run_all.tcl" -mastercpus $cpu
    fi

    echo "Returning to base directory"

    cd ../..
}
echo "run_gen var1=block var2=directory on block var3=script_name"
echo "block = block_1 / block_2 / block_3 /block_4"
echo "directory = folder name on destination"
echo "script_name  = script name from sets_folder"
```

Appendix 19: Script *report_parser.tcl*

```
#General Procedure for report parsing:

proc report_parser {args} {

   parse_proc_arguments -args $args results

   #Input proc arguments
   set global_power_flag              [info exists results(-global_power)]
   set clock_power_flag               [info exists results(-clock_power)]
   set latency_flag                   [info exists results(-latency)]
   set general_routing_flag           [info exists results(-general_routing)]
   set local_skew_flag                [info exists results(-local_skew)]
   set wirelength_flag                [info exists results(-wirelength)]
   set congestion_flag                [info exists results(-layer_congestion)]
   set unit_flag                      [info exists results(-units)]
   set utilization_flag               [info exists results(-utilization)]
   set clk_area_flag                  [info exists results(-clk_area)]
   set power_flag                     [info exists results(-power_scenario)]
   set timing_flag                    [info exists results(-timing_scenarios)]
   set report_flag                    [info exists results(-report_flag)]
   set compact_flag                   [info exists results(-compact)]
   set QoR_file_flag                  [info exists results(-QoR_file)]
   set QoR_reformat_flag              [info exists results(-QoR_reformat)]
   set summary_flag                   [info exists results(-summary)]

   if {[info exists results(-step)]} { ;# If step is defined:
      if {!$report_flag} { ;# If report_flag (i.e) file already exists and reports are not needed again.
         set step_sel $results(-step)
         rec_mkdir reports/$step_sel/selected
      } else {
         set step_sel $results(-step)
      }
   }

   if {[info exists results(-power_scenario)]} {
      set pw_scen $results(-power_scenario)
   } else {
      echo "Power scenario not found. It will be atempted to be obtained from the active scenario
list.\n"
   }

   if {[info exists results(-timing_scenarios)]} {
      set list_timing $results(-timing_scenarios)
   } else {
      echo "Timing scenarios not found. It will be atempted to be obtained from the active scenario
list.\n"
   }

   if {$QoR_file_flag} {
      set QoR_file_name $results(-QoR_file)
   }


   #-tee settings:

   if {[info exists results(-tee)]} {
      set tee "-tee -var"
   } else {
      set tee "-var"
   }
   if {[info exists results(-units)]} {
      set unit $results(-units)
   }
```

```
set nil "~"

# Unit specificatoin

if {$unit_flag} {
   if {string match $unit "ps"} {
      set unit 1000000
   } else {
      set unit 1000
   }
} else {
   catch {redirect -var y {report_units}}

   if {[regexp {(\S+)\s+Second} $y match unit]} {
      if {[regexp {e-12} $unit]} {
         set unit 1000000
      } else {
         set unit 1000
      }
   } elseif {[regexp {ns} $y]} {
      set unit 1000
   } elseif {[regexp {ps} $y]} {
      set unit 1000000
   }
}


#Layer obtention

set LAYERS_CONGETION [get_attr [get_layers -filter "layer_type == interconnect"] name]


if {$::synopsys_program_name == "icc2_shell"} {
   if {$global_power_flag} {

      echo -n "Running Global Power Report\n"

      #Obtains all the power scenarios and puts them on a list called $power_scenario
      if {$power_flag} {
         set power_scenario $pw_scen
         echo "Power scenario found\n"
      } else {

         set list1 [get_object_name [all_scenarios]]

         foreach list_scenario $list1 {
            if {[regexp {(?=power)} $list_scenario]} {
               lappend power_scenario $list_scenario
            }
         }
         foreach scenario $power_scenario {
            regsub $scenario $list1 "" list1
         }
         echo "Power scenarios from active scenario list used\n"
      }

      foreach pow_scen $power_scenario {

         if {$report_flag} {
            set text_file [open "reports/${step_sel}/selected/report_power_${pow_scen}.rpt" "r"]
            set x [read $text_file]
         } else {
            redirect {*}$tee x {report_power -nosplit -scenario $pow_scen}
            echo "$x" > "reports/${step_sel}/selected/report_power_${pow_scen}.rpt"
```

```
            }

            set group_power_just_set 0

            foreach line [split  $x "\n"] {
               if {[regexp {^\s*io_pad\s*(\S+)\s*(\S+)\s*(\S+)\s*(\S+)\s*(\S+)\s*(\S+)\s*$} $line match internal
switching leakage total a b]} {
                  set group_perc(io_pad) [concat $a$b]
                  set group_int(io_pad) $internal
                  set group_switch(io_pad) $switching
                  set group_leak(io_pad) $leakage
                  set group_total(io_pad) $total
               } elseif {[regexp {^\s*memory\s*(\S+)\s*(\S+)\s*(\S+)\s*(\S+)\s*(\S+)\s*(\S+)\s*$} $line match
internal switching leakage total a b]} {
                  set group_perc(mem) [concat $a$b]
                  set group_int(mem) $internal
                  set group_switch(mem) $switching
                  set group_leak(mem) $leakage
                  set group_total(mem) $total
               } elseif {[regexp {^\s*black_box\s*(\S+)\s*(\S+)\s*(\S+)\s*(\S+)\s*(\S+)\s*(\S+)\s*$} $line match
internal switching leakage total a b]} {
                  set group_perc(bb) [concat $a$b]
                  set group_int(bb) $internal
                  set group_switch(bb) $switching
                  set group_leak(bb) $leakage
                  set group_total(bb) $total
               } elseif {[regexp {^\s*clock_network\s*(\S+)\s*(\S+)\s*(\S+)\s*(\S+)\s*(\S+)\s*(\S+)\s*$} $line
match internal switching leakage total a b]} {
                  set group_perc(clk) [concat $a$b]
                  set group_int(clk) $internal
                  set group_switch(clk) $switching
                  set group_leak(clk) $leakage
                  set group_total(clk) $total
               } elseif {[regexp {^\s*register\s*(\S+)\s*(\S+)\s*(\S+)\s*(\S+)\s*(\S+)\s*(\S+)\s*$} $line match
internal switching leakage total a b]} {
                  set group_perc(reg) [concat $a$b]
                  set group_int(reg) $internal
                  set group_switch(reg) $switching
                  set group_leak(reg) $leakage
                  set group_total(reg) $total
               } elseif {[regexp {^\s*sequential\s*(\S+)\s*(\S+)\s*(\S+)\s*(\S+)\s*(\S+)\s*(\S+)\s*$} $line match
internal switching leakage total a b]} {
                  set group_perc(seq) [concat $a$b]
                  set group_int(seq) $internal
                  set group_switch(seq) $switching
                  set group_leak(seq) $leakage
                  set group_total(seq) $total
               } elseif {[regexp {^\s*combinational\s*(\S+)\s*(\S+)\s*(\S+)\s*(\S+)\s*(\S+)\s*(\S+)\s*$} $line
match internal switching leakage total a b]} {
                  set group_perc(comb) [concat $a$b]
                  set group_int(comb) $internal
                  set group_switch(comb) $switching
                  set group_leak(comb) $leakage
                  set group_total(comb) $total

                  set group_power_just_set 1
               } elseif {[regexp {^\s*Total\s*(\S+)\s*(\S+)\s*(\S+)\s*(\S+)\s*(\S+)\s*(\S+)\s*(\S+)\s*(\S+)\s*$}
$line match internal a switching b leakage c total d]} {
                  if {$group_power_just_set} {
                     set group_perc(tot) $nil
                     set group_int(tot) [concat $internal$a]
                     set group_switch(tot) [concat $switching$b]
                     set group_leak(tot) [concat $leakage$c]
                     set group_total(tot) [concat $total$d]
```

```
                    set internal_total [concat $internal$a]
                    set switching_total [concat $switching$b]
                    set leakage_total [concat $leakage$c]
                    set total_total [concat $total$a]
                    set group_power_just_set 0
                }
            }
        }
    }
    if {$clock_power_flag} { #OK ATM, only has 1 clock/scenario

        if {$power_flag} {
            set power_scenario $pw_scen
        } else {

            set list1 [get_object_name [all_scenarios]]
            foreach list_scenario $list1 {
                if {[regexp {(?=power)} $list_scenario]} {
                    lappend power_scenario $list_scenario
                }
            }
            foreach scenario $power_scenario {
                regsub $scenario $list1 "" list1
            }
        }


        echo -n "Running Clock Power Report\n"

        foreach pow_scen $power_scenario {
            if {$report_flag} {
                set text_file [open "reports/${step_sel}/selected/clock_QoR_power_${pow_scen}.rpt" "r"]
                set x [read $text_file]
            } else {
                redirect {*}$tee x {report_clock_QoR -type power -nosplit -scenario $pow_scen}
                echo "$x" > "reports/${step_sel}/selected/clock_QoR_power_${pow_scen}.rpt"
            }

            foreach line [split $x "\n"] {
                if {[regexp {^\s*Total\s*(\S+)\s*(\S+)\s*(\S+)\s*(\S+)\s*(\S+)\s*$} $line match clk_leakage
clk_internal sink_internal net_switching total_dyn total]} {
                    set clk_clk_leakage $clk_leakage
                    set clk_clk_internal $clk_internal
                    set clk_sink_internal $sink_internal
                    set clk_net_switching $net_switching
                    set clk_total_dyn $total_dyn
                    set clk_total $total
                }
            }

        }

    }

    if {$general_routing_flag} { ; #OK

        echo -n "Running General Routing Report\n"

        if {$report_flag} {
            set text_file [open "reports/${step_sel}/selected/check_route.rpt" "r"]
            set x [read $text_file]
        } else {
            redirect {*}$tee x {check_route -antenna true}
            echo "$x" > "reports/${step_sel}/selected/check_route.rpt"
```

```
        }

        set group_general_route_just_set 0
        set gflag 0

        foreach line [split $x "\n"] {
           if {[regexp {^\s*Total number of nets =\s*(\S+)\s*, of which\s*(\S+)\s*are not extracted\s*$} $line
match total_nets not_extracted]} {
              set groute_total_nets $total_nets
           } elseif {[regexp {^\s*Total number of open nets =\s*(\S+)\s*, of which\s*(\S+)\s*are frozen\s*$}
$line match open_nets not_frozen]} {
              set groute_open_nets $open_nets
           } elseif {[regexp {^\s*Total Wire Length =\s*(\S+)\s*micron\s*$} $line match wire_length]} {
              set groute_wire_length $wire_length
           } elseif {[regexp {^\s*Total Number of Contacts =\s*(\S+)\s*$} $line match contacts]} {
              set groute_contacts $contacts
           } elseif {[regexp {^\s*Total Number of Wires =\s*(\S+)\s*$} $line match wires]} {
              set groute_wires $wires
           } elseif {[regexp {^\s*Total Number of Routed Wires =\s*(\S+)\s*$} $line match routed_wires]} {
              set groute_routed_wires $routed_wires
           } elseif {[regexp {^\s*Total Routed Wire Length =\s*(\S+)\s*micron\s*$} $line match
routed_wire_length]} {
              set groute_routed_wire_length $routed_wire_length
           } elseif {[regexp {^\s*Total Number of Routed Contacts =\s*(\S+)\s*$} $line match
routed_contacts]} {
              set groute_routed_contacts $routed_contacts
           } elseif {[regexp {^\s*Total number of DRCs =\s*(\S+)\s*$} $line match DRC]} {
              set groute_DRC $DRC
           } elseif {[regexp {^\s*Total number of antenna violations = \s*(\S+)\s*$} $line match antenna]} {
              set groute_antenna $antenna
           }
        }
        set groute_tot 0
        foreach line [split $x "\n"] {

           foreach layer $LAYERS_CONGETION {

              if {[regexp {^\t*\s*Redundant} $line match]} {
                 set gflag 0
                 continue
              } elseif {[regexp {^\s*DRC-SUMMARY:\s*$} $line match]} {

                 set gflag 1
                 continue

              }
              if {$gflag && [regexp "$layer" $line match]} {

                 if {[regexp {^\t*\s*(\S+)\s*(\S+)\s*(\S+)\s*(\S+)\s*(\S+)\s*$} $line match a layer tp length
micron]} {
                 set groute_length($layer) $length
                 set groute_tot [expr {$groute_tot + $length}]

                 }
              }
           }
        }
     }



     if {$latency_flag} {

        if {$timing_flag} {
```

```
            set list1 $list_timing
            echo "Timing scenarios found.\n"

         } else {
            #First of all obtains all timing scenarios to run the timing_scenario foreach

            set list1 [get_object_name [all_scenarios]]

            foreach list_scenario $list1 {
               if {[regexp {(?=power)} $list_scenario]} {
                  lappend power_scenario $list_scenario
               }
            }
            foreach scenario $power_scenario {
               regsub $scenario $list1 "" list1
            }
         }

         echo -n "Running scenario latency reports\n"

         foreach timing_scenario $list1 {

         if {$report_flag} {
            set text_file [open "reports/${step_sel}/selected/latency_${timing_scenario}.rpt" "r"]
            set x [read $text_file]
         } else {
            redirect {*}$tee x {report_clock_QoR -type latency -show_paths -scenarios $timing_scenario -
nosplit}
            echo "$x" > "reports/${step_sel}/selected/latency_${timing_scenario}.rpt"
         }

            foreach line [split $x "\n"] {

               if {[regexp {^\s*Largest Path #1\s*$} $line match path_num]} {
                  set path_1_set 1
                  continue
               } elseif {[regexp {^\s*Mode\s*:\s*(\S+)\s*$} $line match mode_1]} {

                  if {$path_1_set} {
                     set group_modeL($timing_scenario) $mode_1
                     set group_pathL($timing_scenario) "Large"
                  }
               } elseif {[regexp {^\s*Corner\s*:\s*(\S+)\s*$} $line match corner_1]} {
                  if {$path_1_set} {
                     set group_cornerL($timing_scenario) $corner_1

                  }
               } elseif {[regexp {^\s*Scenario\s*:\s*(\S+)\s*$} $line match scenario_1]} {
                  if {$path_1_set} {
                     set group_scenarioL($timing_scenario) $scenario_1
                  }
               } elseif {[regexp {^\s*Latency\s*:\s*(\S+)\s*$} $line match latency_1]} {
                  if {$path_1_set} {
                     set group_latencyL($timing_scenario) $latency_1
                     set path_1_set 0
                  }
               }
            }


         }
      }

      if {$latency_flag} {

         if {$timing_flag} {
```

```
            set list1 $list_timing
            echo "Timing scenarios found.\n"

         } else {
            #First of all obtains all timing scenarios to run the timing_scenario foreach

            set list1 [get_object_name [all_scenarios]]

            foreach list_scenario $list1 {
               if {[regexp {(?=power)} $list_scenario]} {
                  lappend power_scenario $list_scenario
               }
            }
            foreach scenario $power_scenario {
               regsub $scenario $list1 "" list1
            }
         }

         echo -n "Running scenario latency reports\n"

         foreach timing_scenario $list1 {

         if {$report_flag} {
            set text_file [open "reports/${step_sel}/selected/latency_${timing_scenario}.rpt" "r"]
            set x [read $text_file]
         } else {
            redirect {*}$tee x {report_clock_QoR -type latency -show_paths -scenarios $timing_scenario -
nosplit}
            echo "$x" > "reports/${step_sel}/selected/latency_${timing_scenario}.rpt"
         }

            foreach line [split $x "\n"] {

               if {[regexp {^\s*Smallest Path #1\s*$} $line match path_num]} {
                  set path_1_set 1
                  continue
               } elseif {[regexp {^\s*Mode\s*:\s*(\S+)\s*$} $line match mode_1]} {

                  if {$path_1_set} {
                     set group_modeS($timing_scenario) $mode_1
                     set group_pathS($timing_scenario) "Small"
                  }
               } elseif {[regexp {^\s*Corner\s*:\s*(\S+)\s*$} $line match corner_1]} {
                  if {$path_1_set} {
                     set group_cornerS($timing_scenario) $corner_1
                  }
               } elseif {[regexp {^\s*Scenario\s*:\s*(\S+)\s*$} $line match scenario_1]} {
                  if {$path_1_set} {
                     set group_scenarioS($timing_scenario) $scenario_1
                  }
               } elseif {[regexp {^\s*Latency\s*:\s*(\S+)\s*$} $line match latency_1]} {
                  if {$path_1_set} {
                     set group_latencyS($timing_scenario) $latency_1
                     set path_1_set 0
                  }
               }
            }


         }
      }

      if {$utilization_flag} {
         echo -n "Running utilization report\n"

         if {$report_flag} {
```

```
            set text_file [open "reports/${step_sel}/selected/utilization.rpt" "r"]
            set x [read $text_file]
         } else {
            redirect {*}$tee x {report_utilization}
            echo "$x" > "reports/${step_sel}/selected/utilization.rpt"
         }

         foreach line [split $x "\n"] {

            if {[regexp {^\s*Utilization Ratio:\s*(\S+)\s*$} $line match u_ratio]} {
               set utilization $u_ratio
            } elseif {[regexp {^\s*Total Area:\s*(\S+)\s*$} $line match area]} {
               set total_area $area
            } elseif {[regexp {^\s*Total Capacity Area:\s*(\S+)\s*$} $line match area]} {
               set total_cap_area $area
            } elseif {[regexp {^\s*Total Area of cells:\s*(\S+)\s*$} $line match area]} {
               set total_cell_area $area

            } elseif {[regexp {^\s*- hard_macros\s*:\s*(\S+)\s*$} $line match area]} {
               set hard_macros_area $area
            } elseif {[regexp {^\s*- macro_keepouts\s*:\s*(\S+)\s*$} $line match area]} {
               set macro_keepouts_area $area
            } elseif {[regexp {^\s*- soft_macros\s*:\s*(\S+)\s*$} $line match area]} {
               set soft_macros_area $area
            } elseif {[regexp {^\s*- io_cells\s*:\s*(\S+)\s*$} $line match area]} {
               set io_cells_area $area
            } elseif {[regexp {^\s*- hard_blockages\s*:\s*(\S+)\s*$} $line match area]} {
               set hard_blockages_area $area
            }
         }
      }

      if {$clk_area_flag} { ;#Must be adapted for multiple clocks

         echo -n "Running clock area report\n"

         if {$report_flag} {
            set text_file [open "reports/${step_sel}/selected/clock_area.rpt" "r"]
            set x [read $text_file]
         } else {
            redirect {*}$tee x {report_clock_QoR -type area}
            echo "$x" > "reports/${step_sel}/selected/clock_area.rpt"
         }

         foreach line [split $x "\n"] {

            if {[regexp {^\s*Clock\s*Sinks\s*Clock\s*} $line]} {
            set clk_area_just_set 1

            continue
            } elseif {[regexp {^\s*Total\s*(\S+)\s*(\S+)\s*(\S+)\s*(\S+)\s*(\S+)\s*(\S+)\s*(\S+)\s*(\S+)\s*$} $line
match sinks cell_count stdcell_area repeater_count repeater_area physical_sinks sink_area macro_area]} {

               if {$clk_area_just_set} {
                  set clk_sinks $sinks
                  set clk_cell_count $cell_count
                  set clk_stdcell_area $stdcell_area
                  set clk_repeater_count $repeater_count
                  set clk_repeater_area $repeater_area
                  set clk_physical_sinks $physical_sinks
                  set clk_sink_area $sink_area
                  set clk_macro_area $macro_area
                  set clk_area_just_set 0
               }
            }
         }
```

```
    }
    if {$wirelength_flag} {
        echo -n "Running wirelength report\n"

        if {$report_flag} {
            set text_file [open "reports/${step_sel}/selected/design_wirelength.rpt" "r"]
            set x [read $text_file]
        } else {
            redirect {*}$tee x {report_design -routing}
            echo "$x" > "reports/${step_sel}/selected/design_wirelength.rpt"
        }

        set signal_wire 0
        set clock_wire 0
        set pg_wire 0
        set hv_wire 0
        set sflag 0

        set llist {}

        foreach line [split $x "\n"] {

        if {[regexp {^\s*Total wire length =\s*(\S+)\s*micron\s*$} $line match wire_length]} {
            set route_wirelength $wire_length
        } elseif {[regexp {^\s*Total number of wires =\s*(\S+)\s*$} $line match wire_number]} {
            set route_wire $wire_number
        } elseif {[regexp {^\s*Total number of contacts =\s*(\S+)\s*$} $line match contact_number]} {
            set route_contact $contact_number
        } elseif {[regexp {^\s*Clock wiring statistics\s*$} $line match]} {
            set sflag 0
        } elseif {[regexp {^\s*Signal Wiring Statistics\s*$} $line match]} {
            set sflag 1
            continue


        } elseif {$sflag && [regexp {^\s*(\S+)\s*(\S+)\s*(\S+)\s*(\S+)\s*(\S+)\s*$} $line match layer numwires
percent_totalnumwires wirelength percent_totallength]} {
            lappend llist $layer
        }

}
        set tot_wire_s 0
        set tot_wire_c 0
        set tot_wire_pg 0
        set tot_wire_hv 0

        foreach layer $llist {

            foreach line [split $x "\n"] {

                if {[regexp {^\s*FINAL WIRING STATISTICS\s*$} $line match]} {
                    set signal_wire 1
                    set ll "s"
                    continue
                } elseif {$signal_wire && [regexp "$layer" $line match]} {
                    if {[regexp {^\s*(\S+)\s*(\S+)\s*(\S+)\s*(\S+)\s*(\S+)\s*$} $line match layer numwires
percent_totalnumwires wirelength percent_totallength]} {
                        set group_numwire([concat $layer$ll]) $numwires
                        set group_totwire([concat $layer$ll]) $percent_totalnumwires
                        set group_wirelength([concat $layer$ll]) $wirelength
                        set group_totlength([concat $layer$ll]) $percent_totallength
                        set tot_wire_s [expr {$tot_wire_s + $wirelength}]
                        set signal_wire 0
                    }
                } elseif {[regexp {^\s*Clock wiring statistics\s*$} $line match]} {
                    set clock_wire 1
```

```
                    set ll "c"
                    continue ; # Sets a flag for clock wiring statistics.
                } elseif {$clock_wire && [regexp "$layer" $line match]} {
                    if {[regexp {^\s*(\S+)\s*(\S+)\s*(\S+)\s*(\S+)\s*(\S+)\s*$} $line match layer numwires
percent_totalnumwires wirelength percent_totallength]} {
                        set group_numwire([concat $layer$ll]) $numwires
                        set group_totwire([concat $layer$ll]) $percent_totalnumwires
                        set group_wirelength([concat $layer$ll]) $wirelength
                        set group_totlength([concat $layer$ll]) $percent_totallength
                        set tot_wire_c [expr {$tot_wire_c + $wirelength}]
                        set clock_wire 0
                    }
                } elseif {[regexp {^\s*P/G wiring statistics\s*$} $line match]} {
                    set pg_wire 1
                    set ll "pg"
                    continue ; # Sets a flag for clock wiring statistics.
                } elseif {$pg_wire && [regexp "$layer" $line match]} {
                    if {[regexp {^\s*(\S+)\s*(\S+)\s*(\S+)\s*(\S+)\s*(\S+)\s*$} $line match layer numwires
percent_totalnumwires wirelength percent_totallength]} {
                        set group_numwire([concat $layer$ll]) $numwires
                        set group_totwire([concat $layer$ll]) $percent_totalnumwires
                        set group_wirelength([concat $layer$ll]) $wirelength
                        set group_totlength([concat $layer$ll]) $percent_totallength
                        set tot_wire_pg [expr {$tot_wire_pg + $wirelength}]
                        set pg_wire 0
                    }
                } elseif {[regexp {^\s*Horizontal/Vertical Wire Distribution\s*$} $line match]} {
                    set hv_wire 1
                    set ll "hv"
                    continue ; # Sets a flag for clock wiring statistics.
                } elseif {$hv_wire && [regexp "$layer" $line match]} {
                    if {[regexp {^\s*(\S+)\s*(\S+)\s*(\S+)\s*(\S+)\s*(\S+)\s*$} $line match layer numwires
percent_totalnumwires wirelength percent_totallength]} {
                        set group_numwire([concat $layer$ll]) $numwires
                        set group_totwire([concat $layer$ll]) $percent_totalnumwires
                        set group_wirelength([concat $layer$ll]) $wirelength
                        set group_totlength([concat $layer$ll]) $percent_totallength
                        set tot_wire_hv [expr {$tot_wire_hv + $wirelength}]
                        set hv_wire 0
                    }
                }
            }
        }

    }
    if {$congestion_flag} {


    set sum_tot 0
    set sum_max 0
    set gmax_of 0

        if {$report_flag} {
            set text_file [open "reports/${step_sel}/selected/layers_congestion.rpt" "r"]
            set x [read $text_file]
        } else {
            redirect {*}$tee x {report_congestion -layers $LAYERS_CONGETION}
            echo "$x" > "reports/${step_sel}/selected/layers_congestion.rpt"
        }

        #Using $LAYERS_CONGESTION

        foreach line [split $x "\n"] {
            foreach layer $LAYERS_CONGESTION {
                if {[regexp "$layer" $line match]} {
```

```
                if {[regexp
{^\s*(\S+)\s*(\S+)\s*(\S+)\s*(\S+)\s*(\S+)\s*(\S+)\s*(\S+)\s*(\S+)\s*(\S+)\s*(\S+)\s*(\S+)\s*$} $line match layer
z tot_of x max_of y per_grc_of a b w grc_max_of]} {
                    set group_tot_of($layer) $tot_of
                    set group_max_of($layer) $max_of
                    set group_perc_grc_of($layer) $per_grc_of
                    set group_perc($layer) [concat $a$b]
                    set group_grc_max_of($layer) $grc_max_of
                    set sum_tot [expr {$tot_of + $sum_tot}]
                    set sum_max [expr {$group_max_of($layer) + $sum_max}]
                    if {$group_max_of($layer) > $gmax_of} {
                        set gmax_of $group_max_of($layer)
                    }
                    continue
                    set sum_tot [expr {$tot_of + $sum_tot}]

                    set sum_max [expr {$group_max_of($layer) + $sum_max}]
                    if {$group_max_of($layer) > $gmax_of} {
                        set gmax_of $group_max_of($layer)
                    }
                }
            }
        }
    }

if {$local_skew_flag} {

    #If the timing_flag is set to 1, timing scenarios are given when calling the proc and are used for the reports.
    #Else it will get the scenarios following a regular expression.

    if {$timing_flag} {
        set list1 $list_timing
        echo "Timing scenarios found.\n"

    } else {
        #First of all obtains all timing scenarios to run the timing_scenario foreach

        set list1 [get_object_name [all_scenarios]]

        foreach list_scenario $list1 {
            if {[regexp {(?=power)} $list_scenario]} {
                lappend power_scenario $list_scenario
            }
        }
        foreach scenario $power_scenario {
            regsub $scenario $list1 "" list1
        }
    }
    set flag_lskew 0
    set lskew_flag_init 1
    set list_clks {}

    foreach timing_scenario $list1 {
    set lskew_flag_init 1
    set flag_lskew 0
    #If the report flag is set to 1 (i.e. report file already exists on the given directory), it accesses the report file,
saves it on a variable and uses it to parse and generate csv file.
        if {$report_flag} {
            set text_file [open "reports/${step_sel}/selected/local_skew_${timing_scenario}.rpt" "r"]
            set x [read $text_file]
        } else {
            redirect {*}$tee x {report_clock_QoR -type local_skew -nosplit -scenarios $timing_scenario}
            echo "$x" > "reports/${step_sel}/selected/local_skew_${timing_scenario}.rpt"
        }
```

```
      foreach line [split $x "\n"] {  ;# If All Clocks Line has not been
         if {$lskew_flag_init} {
            if {[regexp {^\s*### Mode:\s*(\S+),\s*Scenario:\s*(\S+)\s*$} $line match mode scenario]} {
            set lskew_mode($scenario) $mode
            set lskew_scenario($scenario) $scenario
            set flag_lskew 1
            continue
            } elseif {$flag_lskew} {
               if {[regexp {^\s*--------------------------------} $line match]} {
                  set flag_lskew 0
                  set lskew_flag_init 0
               } elseif {[regexp {^\s*(\S+)\s*(\S+)\s*,\s*(\S+)\s*(\S+)\s*(\S+)\s*(\S+)\s*(\S+)\s*(\S+)\s*(\S+)\s*$}
$line match clock a b sinks gskew maxlat locpair maxsetup maxhold]} {
                  set lskew_clock([concat $scenario$clock]) $clock
                  set attrs [concat $a$b]
                  set lskew_attrs([concat $scenario$clock]) $attrs
                  set lskew_sinks([concat $scenario$clock]) $sinks
                  set lskew_gskew([concat $scenario$clock]) $gskew
                  set lskew_maxlat([concat $scenario$clock]) $maxlat
                  set lskew_paircount([concat $scenario$clock]) $locpair
                  set lskew_maxsetup([concat $scenario$clock]) $maxsetup
                  set lskew_maxhold([concat $scenario$clock]) $maxhold
                  lappend list_clks $clock
               }
            }
         }
      }

   }
}
   } ;#If program selection


   if {$global_power_flag} {

      #Power reports csv file generation
      echo "Power reports csv file generation\n"

      set csv_file "reports/${step_sel}/selected/QoR_power.csv"
      set csv [open $csv_file "w"]

      set bar "----------------"

      set list_gen_power {io_pad mem bb clk reg seq comb tot}

      puts $csv "sep=,"

      puts $csv "General Power Consumption\n"

      puts $csv "Power Group, Percentage Power, Internal Power, Switching Power, Leakage Power,
Total Power\n"

      foreach g $list_gen_power {
         puts $csv "$g, $group_perc($g), $group_int($g), $group_switch($g), $group_leak($g),
$group_total($g)\n"
      }


      close $csv

   } elseif {$clock_power_flag} {

      echo "Clock power csv file generation\n"
```

```
    set csv_file "reports/${step_sel}/selected/QoR_power.csv"
    set csv [open $csv_file "w"]

    puts $csv "sep=,"

    puts $csv "Clock Power Consumption\n"

    puts $csv "Leakage Power, Internal Power, Internal Sink Power, Net Switching Power, Total
Dynamic Power, Total Power\n"

    puts $csv "$clk_clk_leakage, $clk_clk_internal, $clk_sink_internal, $clk_net_switching,
$clk_total_dyn, $clk_total\n"

    close $csv

  }

  if {$global_power_flag && $clock_power_flag} {
    echo "Power reports csv file generation\n"


    set csv_file "reports/${step_sel}/selected/QoR_power.csv"
    set csv [open $csv_file "w"]

    puts $csv "sep=,"

    puts "General Power Consumption\n"
    puts $csv "Power Group, Percentage power, Internal Power, Switching Power, Leakage Power,
Total Power\n"

    foreach g $list_gen_power {
      puts $csv "$g, $group_perc($g), $group_int($g), $group_switch($g), $group_leak($g),
$group_total($g)\n"
    }
    puts $csv "$bar\n"

    puts $csv "Leakage Power, Internal Power, Internal Sink Power, Net Switching Power, Total
Dynamic Power, Total Power\n"
    puts $csv "$clk_clk_leakage, $clk_clk_internal, $clk_sink_internal, $clk_net_switching,
$clk_total_dyn, $clk_total\n"

    close $csv

  }
  if {$general_routing_flag} {

    echo "Routing information csv file generation\n"

    set nil "~"

    if {![info exists total_nets]} {set total_nets $nil}
    if {![info exists open_nets]} {set open_nets $nil}
    if {![info exists wire_length]} {set wire_length $nil}
    if {![info exists contacts]} {set contacts $nil}
    if {![info exists wires]} {set wires $nil}
    if {![info exists routed_wires]} {set routed_wires $nil}
    if {![info exists routed_wire_length]} {set routed_wire_length $nil}
    if {![info exists routed_contacts]} {set routed_contacts $nil}
    if {![info exists groute_DRC]} {set groute_DRC $nil}
    if {![info exists groute_antenna]} {set groute_antenna $nil}

    set csv_file "reports/${step_sel}/selected/QoR_routing.csv"
    set csv [open $csv_file "w"]

    puts $csv "sep=,"
```

```
      puts $csv "General information\n"
      puts $csv "Total Nets, Total Open Nets, Total Wirelength, Total Contacts, Total Wires, Total
Routed Wires, Total Routed Wirelength, Total Routed Contacts, Total DRC, Total Antenna
Violations\n"
      puts $csv "$total_nets, $open_nets, $wire_length, $contacts, $wires, $routed_wires,
$routed_wire_length, $routed_contacts, $groute_DRC, $groute_antenna\n"

      puts $csv "Layer information\n"

      foreach g $LAYERS_CONGETION {
         puts $csv "$g, $groute_length($g), microns"
      }


      close $csv
   }

   if {$latency_flag} {

      set bar "---------"
      echo "Latency Information csv file generation\n"

      set csv_file "reports/${step_sel}/selected/QoR_latency.csv"
      set csv [open $csv_file "w"]

      puts $csv "sep=,"

      puts $csv "Mode, Corner, Scenario, Latency, Path Type\n"
      foreach timing_scenario $list1 {
         puts $csv "$group_modeL($timing_scenario), $group_cornerL($timing_scenario),
$group_scenarioL($timing_scenario), $group_latencyL($timing_scenario),
$group_pathL($timing_scenario)"
      }

      puts $csv "$bar\n"
      foreach timing_scenario $list1 {
         puts $csv "$group_modeS($timing_scenario), $group_cornerS($timing_scenario),
$group_scenarioS($timing_scenario), $group_latencyS($timing_scenario),
$group_pathS($timing_scenario)"
      }

      close $csv


   }

   if {$clk_area_flag} {
      echo "Clock area information csv file generation\n"

      set csv_file "reports/${step_sel}/selected/QoR_area.csv"
      set csv [open $csv_file "w"]

      puts $csv "sep=,"

      puts $csv "Clock Sinks, Clock Cell Count, Standard Cell Area, Repeater Count, Repeater Area,
Physical Sinks, Sinks Area, Macro Area\n"
      puts $csv "$clk_sinks, $cell_count, $stdcell_area, $repeater_count, $repeater_area,
$physical_sinks, $sink_area, $macro_area\n"

      close $csv
   }

   if {$utilization_flag} {
      echo "Utilization information csv file generation\n"

      set csv_file "reports/${step_sel}/selected/QoR_area.csv"
```

```
    set csv [open $csv_file "w"]

    puts $csv "sep=,"

    puts $csv "Utilization Ratio, Total Area, Total Capacity Area, Total Area of Cells, Hard Macros,
Macro Keepouts, Soft Macros, IO Cells, Hard Blockages\n"
    puts $csv "$utilization, $total_area, $total_cap_area, $total_cell_area, $hard_macros_area,
$macro_keepouts_area, $soft_macros_area, $io_cells_area, $hard_blockages_area\n"

    close $csv
  }

  if {$utilization_flag && $clk_area_flag} {
    echo  "Utilization and clock area csv file generation\n"

    set bar "----------------------"

    set csv_file "reports/${step_sel}/selected/QoR_area.csv"
    set csv [open $csv_file "w"]

    puts $csv "sep=,"

    puts $csv "Utilization Ratio, Total Area, Total Capacity Area, Total Area of Cells, Hard Macros,
Macro Keepouts, Soft Macros, IO Cells, Hard Blockages\n"
    puts $csv "$utilization, $total_area, $total_cap_area, $total_cell_area, $hard_macros_area,
$macro_keepouts_area, $soft_macros_area, $io_cells_area, $hard_blockages_area\n"
    puts $csv "$bar\n"

    puts $csv "Clock Sinks, Clock Cell Count, Standard Cell Area, Repeater Count, Repeater Area,
Physical Sinks, Sinks Area, Macro Area\n"
    puts $csv "$clk_sinks, $cell_count, $stdcell_area, $repeater_count, $repeater_area,
$physical_sinks, $sink_area, $macro_area\n"

    close $csv

  }

  if {$wirelength_flag} {



    echo "Wirelength csv file generation\n"
    set csv_file "reports/${step_sel}/selected/QoR_wirelength.csv"
    set csv [open $csv_file "w"]

    puts $csv "sep=,"
    puts $csv "Signal Wiring Statistics\n"
    puts $csv "Metal Layer, Num wires, % of total#, Wire length, % of total length\n"

    set ll "s"

    foreach g $llist {

      puts $csv "$g, $group_numwire([concat $g$ll]), $group_totwire([concat $g$ll]),
$group_wirelength([concat $g$ll]), $group_totlength([concat $g$ll])\n"
    }

    puts $csv "Clock Wiring Statistics\n"
    puts $csv "Metal Layer, Num wires, % of total#, Wire length, % of total length\n"

    set ll "c"

    foreach g $llist {

      puts $csv "$g, $group_numwire([concat $g$ll]), $group_totwire([concat $g$ll]),
$group_wirelength([concat $g$ll]), $group_totlength([concat $g$ll])\n"
```

```
    }

    puts $csv "P/G Wiring Statistics\n"
    puts $csv "Metal Layer, Num wires, % of total#, Wire length, % of total length\n"

    set ll "pg"

    foreach g $llist {

      puts $csv "$g, $group_numwire([concat $g$ll]), $group_totwire([concat $g$ll]),
$group_wirelength([concat $g$ll]), $group_totlength([concat $g$ll])\n"
    }

    puts $csv "H/V Wiring Statistics\n"
    puts $csv "Metal Layer, Num wires, % of total#, Wire length, % of total length\n"

    set ll "hv"

    foreach g $llist {

      puts $csv "$g, $group_numwire([concat $g$ll]), $group_totwire([concat $g$ll]),
$group_wirelength([concat $g$ll]), $group_totlength([concat $g$ll])\n"
    }

    close $csv
  }

  if {$congestion_flag} {


    echo "Congestion csv file generation\n"
    set csv_file "reports/${step_sel}/selected/QoR_congestion.csv"
    set csv [open $csv_file "w"]

    puts $csv "sep=,"
    puts $csv "Layer congestion\n"
    puts $csv "Layer Name, Total Overflow, Max Overflow, #GRCs has overflow, (%), #GRCs has max
overflow\n"

    foreach g $LAYERS_CONGETION {
      puts $csv "$g, $group_tot_of($g), $group_max_of($g), $group_perc_grc_of($g),
$group_perc($g), $group_grc_max_of($g)"
    }
    close $csv
  }

  if {$local_skew_flag} {

    echo "Local skew csv file generation\n"

    foreach element $list_clks {dict set tmp $element 1}
    set hh [dict keys $tmp]

    set nil "~"
    foreach g $list1 {
      foreach h $hh {
      if {![info exists lskew_clock([concat $g$h])]} {set lskew_clock([concat $g$h]) $nil}
      if {![info exists lskew_attrs([concat $g$h])]} {set lskew_attrs([concat $g$h]) $nil}
      if {![info exists lskew_sinks([concat $g$h])]} {set lskew_sinks([concat $g$h]) $nil}
      if {![info exists lskew_gskew([concat $g$h])]} {set lskew_gskew([concat $g$h]) $nil}
      if {![info exists lskew_maxlat([concat $g$h])]} {set lskew_maxlat([concat $g$h]) $nil}
      if {![info exists lskew_paircount([concat $g$h])]} {set lskew_paircount([concat $g$h]) $nil}
      if {![info exists lskew_maxsetup([concat $g$h])]} {set lskew_maxsetup([concat $g$h]) $nil}
      if {![info exists lskew_maxhold([concat $g$h])]} {set lskew_maxhold([concat $g$h]) $nil}
      }
    }
```

```
    set csv_file "reports/${step_sel}/selected/QoR_lskew.csv"
    set csv [open $csv_file "w"]

    puts $csv "sep=,"
    puts $csv "Local skew\n"



    foreach g $list1 {
        puts $csv "Clock Name, Attributes, Number of Sinks, Global Skew, Max Latency, Local skew
Pair Count, Max Setup Local skew, Max Hold Local skew\n"
        foreach h $hh {
        puts $csv "$lskew_clock([concat $g$h]), $lskew_attrs([concat $g$h]), $lskew_sinks([concat
$g$h]), $lskew_gskew([concat $g$h]), $lskew_maxlat([concat $g$h]), $lskew_paircount([concat
$g$h]), $lskew_maxsetup([concat $g$h]), $lskew_maxhold([concat $g$h])\n"
        }
        puts $csv "Mode is $lskew_mode($g), Scenario is $lskew_scenario($g)\n"
        puts $csv "----------\n"


    }
    close $csv
  }

  if {$compact_flag} {

    echo "Compact reporting\n"

    set csv_file "reports/${step_sel}/selected/QoR_compact.csv"
    set csv [open $csv_file "w"]


    puts $csv "sep=,"
    puts $csv "Scenario, Utilization Ratio, Total Area of Cells, Clock Cell Count, Standard Cell Area,
Repeater Count, Repeater Area, Memory Power, Clock Power, Register power, Sequential Power,
Combinational Power, Total Power, Clock Leakage, Clock Internal, Clock Internal Sink, Clock
Switching, Clock Dynamic, Clock Total, Total Overflow, Max Overflow (SL), Max Overflow (AL), Signal
Wiring, Clock Wiring, H/V Wiring, Total Wirelength, Total Contacts, Total Wires, Total Routed Contact,
Total DRC, Total Routing (L), Global Skew, Max Setup Local skew, Max Hold Local skew, Latency (L),
Latency (S)"

    foreach timing_scenario $list1 {
        puts $csv "$timing_scenario, $utilization, $total_cell_area, $cell_count, $stdcell_area,
$repeater_count, $repeater_area, $group_total(mem), $group_total(clk), $group_total(reg),
$group_total(seq), $group_total(comb), $group_total(tot), $clk_clk_leakage, $clk_clk_internal,
$clk_sink_internal, $clk_net_switching, $clk_total_dyn, $clk_total, $sum_tot, $sum_max, $gmax_of,
$tot_wire_s, $tot_wire_c, $tot_wire_hv, $wire_length, $contacts, $wires, $routed_contacts,
$groute_DRC, $groute_tot, $lskew_gskew([concat $timing_scenario$clock]),
$lskew_maxsetup([concat $timing_scenario$clock]), $lskew_maxhold([concat
$timing_scenario$clock]), $group_latencyL($timing_scenario), $group_latencyS($timing_scenario)"
    }

    close $csv

    echo "Summary reporting \n"

    set worst_latency_long 0
    set worst_latency_short 0
    set worst_gskew 0
    set worst_setup_lskew 0
    set worst_hold_lskew 0


    foreach timing_scenario $list1 {
```

```
        if {$group_latencyL($timing_scenario) > $worst_latency_long} {
            set worst_latency_long $group_latencyL($timing_scenario)
        }

        if {$group_latencyS($timing_scenario) > $worst_latency_short} {
            set worst_latency_short $group_latencyS($timing_scenario)
        }

        if {$lskew_gskew([concat $scenario$clock]) > $worst_gskew} {
            set worst_gskew $lskew_gskew([concat $scenario$clock])
        }

        if {$lskew_maxsetup([concat $timing_scenario$clock]) > $worst_setup_lskew} {
            set worst_setup_lskew $lskew_maxsetup([concat $timing_scenario$clock])
        }

        if {$lskew_maxhold([concat $timing_scenario$clock]) > $worst_hold_lskew} {
            set worst_hold_lskew $lskew_maxhold([concat $timing_scenario$clock])
        }
        echo "lskew_maxhold is $lskew_maxsetup([concat $timing_scenario$clock])"
        echo "Worst hold skew is $worst_setup_lskew"
    }

    set csv_file "reports/${step_sel}/selected/QoR_summary.csv"
    set csv [open $csv_file "w"]

    puts $csv "sep=,"
    puts $csv "Worst Case Summary\n"
    puts $csv "Scenario, Utilization Ratio, Total Area of Cells (um2), Clock Cell Count, Standard Cell
Area (um2), Repeater Count, Repeater Area (um2), Memory Power (uW), Clock Power (uW), Register
power (uW), Sequential Power (uW), Combinational Power (uW), Total Power (uW), Clock Leakage
(uW), Clock Internal (uW), Clock Internal Sink (uW), Clock Switching (uW), Clock Dynamic (uW), Clock
Total (uW), Total Overflow, Max Overflow (SL), Max Overflow (AL), Signal Wiring (um), Clock Wiring
(um), H/V Wiring (um), Total Wirelength (um), Total Contacts, Total Wires, Total Routed Contact, Total
DRC, Total Routing (L) (um), Global Skew (ns), Max Setup Local skew (ns), Max Hold Local skew (ns),
Latency (L) (ns), Latency (S) (ns)"


    puts $csv "---, $utilization, $total_cell_area, $cell_count, $stdcell_area, $repeater_count,
$repeater_area, $group_total(mem), $group_total(clk), $group_total(reg), $group_total(seq),
$group_total(comb), $group_total(tot), $clk_clk_leakage, $clk_clk_internal, $clk_sink_internal,
$clk_net_switching, $clk_total_dyn, $clk_total, $sum_tot, $sum_max, $gmax_of, $tot_wire_s,
$tot_wire_c, $tot_wire_hv, $wire_length, $contacts, $wires, $routed_contacts, $groute_DRC,
$groute_tot, $worst_gskew, $worst_setup_lskew, $worst_hold_lskew, $worst_latency_long,
$worst_latency_short"

    puts $csv "Scenario timing metrics\n"
    puts $csv "Scenario, Global Skew (ns), Max Setup Local skew (ns), Max Hold Local skew (ns),
Latency (L) (ns), Latency (S) (ns)"

    foreach timing_scenario $list1 {
        puts $csv "$timing_scenario, $lskew_gskew([concat $timing_scenario$clock]),
$lskew_maxsetup([concat $timing_scenario$clock]), $lskew_maxhold([concat
$timing_scenario$clock]), $group_latencyL($timing_scenario), $group_latencyS($timing_scenario)"
    }

    close $csv

}

if {$QoR_file_flag} {

    set text_file [open "${QoR_file_name}.csv" "r"]
    set x [read $text_file]

    foreach line [split $x "\n"] {
```

```
        if {[regexp
{^\s*Summary,\s*(\S+),\s*(\S+),\s*(\S+),\s*(\S+),\s*(\S+),\s*(\S+),\s*(\S+),\s*(\S+),\s*(\S+),\s*(\S+),\s*(\S+)\s*$
} $line match q_wns q_skew q_tns q_avgskw q_nvp q_freq q_wnsh q_tnsh q_avgskwh q_nvph]} {
            echo "Data obtained from ${QoR_file_name}.csv - Worst case data across all scenarios\n"
        }
    }
  }

  if {$summary_flag} {

    if {[file exists summary/routeopt_summary.rpt]} {
        set text_file [open "summary/routeopt_summary.rpt" "r"]
        set x [read $text_file]

        foreach line [split $x "\n"] {

            if {[regexp {^\s*(\S+)\s*SETUP INTERNAL FEP\s*(\S+)\s*(\S+)\s*(\S+)\s*$} $line match bar1 bar2
int_setup_fep ]} {echo "Internal Setup FEP: $int_setup_fep\n"}
            if {[regexp {^\s*(\S+)\s*CLOCKSLEW FEP\s*(\S+)\s*(\S+)\s*(\S+)\s*$} $line match bar1 bar2
clockslew_fep]} {echo "CLOCKSLEW FEP: $clockslew_fep\n"}
        }
    }

  }

  if {$QoR_reformat_flag} {

    ### Reformat csv file:
    echo "Reformat .csv file\n"

    #If data from proc_QoR is not available
    if {![info exists q_wns]} {set q_wns "~"}
    if {![info exists q_tns]} {set q_tns "~"}
    if {![info exists int_setup_fep]} {set int_setup_fep "~"}
    if {![info exists clockslew_fep]} {set clockslew_fep "~"}

    set int_int_power [string map {"uW" ""} $group_int(tot)]
    set int_switch_power [string map {"uW" ""} $group_switch(tot)]
    set total_global_power [string map {"uW" ""} $group_total(tot)]
    set global_leakage_power [string map {"uW" ""} $group_leak(tot)]
    set total_dynamic_power [expr { $int_int_power + $int_switch_power}]
    set total_dynamic_power [format "%.4g" $total_dynamic_power]

    set csv_file "reports/${step_sel}/selected/QoR_reformat.csv"
    set csv [open $csv_file "w"]

    puts $csv "sep=,"

    ### Clock reformat data:

    puts $csv "Clock Formatted QoR\n"
    puts $csv "WNS (ns), TNS (ns), Setup FEP, Clockslew FEP, Skew (ns), Latency (ns), Clock Cells,
Clock Repeaters,  Area Clock Cells (um2), Repeater Area (um2), Dynamic Clock Power (uW), Leakage
Clock Power (uW), Total Clock Power (uW)"
    puts $csv "$q_wns, $q_tns, $int_setup_fep, $clockslew_fep, $worst_gskew,
$worst_latency_long, $cell_count, $repeater_count, $stdcell_area, $repeater_area, $clk_total_dyn,
$clk_clk_leakage, $clk_total\n"

    ### General reformat data:

    puts $csv "General Formatted QoR\n"
    puts $csv "Utilization, Total Area (um2), Total Stdcell Area (um2), Total Dynamic Power (uW),
Total Leakage Power (uW), Total Power (uW), Number of DRCs"
    puts $csv "$utilization, $total_area, $total_cell_area, $total_dynamic_power,
$global_leakage_power, $total_global_power, $groute_DRC"
```

```
      close $csv

   }
} ;#If proc

define_proc_attributes report_parser -info "USER PROC: reformats multiple reports" \
   -define_args {
      {-tee              "Optional - displays the output of under-the-hood report_parser command" "" boolean optional}
      {-units            "Optional - specifies the units used" "<ns or ps>" one_of_string {optional value_help {values {ps ns}}}}
      {-global_power     "Optional - specifies if global power information is parsed. Requires active power scenario" "" boolean optional}
      {-clock_power      "Optional - specifies if clock power information is parsed. Requires active power scenario" "" boolean optional}
      {-latency          "Optional - specifies if latency information is parsed. Requires active timing scenarios" "" boolean optional}
      {-general_routing  "Optional - specifies if general routing information is parsed" "" boolean optional}
      {-local_skew       "Optional - specifies if local skew information is parsed. Usable at clockopt onwards Requires active timing scenarios." "" boolean optional}
      {-wirelength       "Optional - specifies if wirelength information is parsed" "" boolean optional}
      {-layer_congestion "Optional - specifies if congestion information is parsed" "" boolean optional}
      {-utilization      "Optional - specifies if the utilization report is used" "" boolean optional}
      {-clk_area         "Optional - specifies if clock area information is used" "" boolean optional}
      {-report_flag      "Optional - specifies wether the reports are generated or parsed from existing" "" boolean optional}
      {-power_scenario   "Optional - specifies the power scenario used for report generation" "" string optional}
      {-timing_scenarios "Optional - specifies the timing scenarios used for report generation" "" list optional}
      {-step             "Required - Must be given to specify the step on which the parsing is made. /reports/$step/selected" "" string required}
      {-compact          "Optional - Makes a compact summary of metrics and prepares it as a csv." "" boolean optional}
      {-QoR_file         "Optional - QoR file obtained from proc_QoR.tcl to add information on WNS/TNS" "" string optional}
      {-QoR_reformat     "Optional - Specifies that the existing data must be reformatted" "" boolean optional}
      {-summary          "Optional - Specifies to extract FEP data from summary files" "" boolean optional}
   }

echo "\treport_parser"
```

Appendix 20: Script *clock_structure_analyser.tcl*

```
### Script to extract information from clock structure:
### Script extracts:
   #-Type of cells and number of each one.
   #-Fanout of intermediates and end drivers.
   #-Manhattan distance between drivers and end-point flops.
   #-Pin capacitance to be obtained from: report_timing -path_type full -nosplit -nets -capacitance -attributes -physical
proc clock_structure_analyzer {args} {
```

```
   parse_proc_arguments -args $args results

   set structure_flag         [info exists results(-structure)]
   set debug_flag             [info exists results(-debugger)]
   set cells_flag             [info exists results(-repeater_cells)]
   set general_analysis_flag      [info exists results(-general_analysis)]
   set endpoint_analysis_flag     [info exists results(-endpoint_analysis)]
   set csv_flag               [info exists results(-csv_file)]
   set detailed_csv_flag          [info exists results(-detailed_csv)]
   set capacitance_analysis_flag      [info exists results(-capacitance_analysis)]

   if {[info exists results(-tee)]} {
      set tee "-tee -var"
   } else {
      set tee "-var"
   }

   if {$cells_flag} {
      set list_clock_cells $results(-repeater_cells)
   }

   if {$::synopsys_program_name == "icc2_shell"} {
      if {$structure_flag} {
         #If the structure flag exists it will use either the selected (if existing) or the existing structure generated
by the reporter.
         if {[file exists "reports/routeopt/selected/clock_QoR_structure.rpt"]} {

            set text_file [open "reports/routeopt/selected/clock_QoR_structure.rpt" "r"]
         } else {

            set text_file [open "reports/routeopt/clock/clock_QoR_structure.rpt" "r"]
         }

         set x [read $text_file]

      } else {
         #If structure flag isnt set, it generates the report and saves it on selected directory
         redirect {*}$tee x {report_clock_QoR -type structure -nosplit}

         echo "$x" > "reports/routeopt/selected/clock_QoR_structure.rpt"
         echo "New structure is is saved on selected directory preserving the existing one\n"
      }

      ### Variable declaration

      set count_source 0 ;# Level zero, source pin
      set count_repeater 0 ;# Repeater intermediate cells
      set count_ICG 0 ;# ICG intermediate cells
      set count_sink 0 ;# Sink Pins
      set count_balance 0 ;# Balance Pins


      ### Fanout calculations

      set max_fanout_rep 0
      set average_fanout_rep 0
      set max_fanout_ICG 0
      set average_fanout_ICG 0

      ### Lists for repeaters and ICGs/Other clock cells

      list repeater_list {dummy}
      list ICG_list

      ### Flags for endpoint detection:
```

```
        set repeater_flag 0
        set ICG_flag 0
        set driver 0
        set endpoint 0


        ### Other variables

        set sep "_" ; #Used to generate the identifier for sinks and balance points
        set repICG_celltype($driver) "unset"
        set current_celltype "unset"
        set line_pointer 0

        if {$general_analysis_flag} {

           foreach line [split $x "\n"] {

              #   Possible Structs to be found:
              #   1) level zero: (Level) Clock Pin [in Port] [Location] [Net] [Fanout]
              #   2) intermediate drivers repeaters: (Level) Clock Pin [Reference cell] [Location] [Net] [Fanout]
              #   3) intermediate drivers ICGs: (Level) Clock Pin [Reference cell] [Location] [Net] [ICG] [Fanout]
              #   4) leafs: (Level) Clock Pin [Reference cell] [Location] [SINK PIN]
              #   5) balance pins: (Level) Clock Pin [Reference cell] [Location] [BALANCE PIN]

              if {[regexp {^\s*(\S+)\s*(\S+)\s*(\S+)in Port(\S+)\s*(\S+)Location (\S+), (\S+)]\s*(\S+)Net:
(\S+)]\s*(\S+)Fanout: (\S+)]\s*$} $line match level pin p1 p2 p3 xloc yloc p4 net p5 fanout]} { ;#Level zero OK
                 set lvlm1 [string map {"(" ""} $level]
                 set lvlm2 [string map {")" ""} $lvlm1]
                 set source_level $lvlm2
                 set source_pin $pin
                 set source_xloc [string map {"(" ""} $xloc]
                 set source_yloc [string map {")" ""} $yloc]
                 set source_net $net
                 set source_fanout $fanout
                 set source_line $line
                 incr count_source

              }

              if {[regexp {^\s*(\S+)\s*(\S+)\s*(\S+)Ref: (\S+)]\s*(\S+)Location (\S+), (\S+)]\s*(\S+)Net:
(\S+)]\s*(\S+)Fanout: (\S+)]\s*$} $line match level pin p1 cell_ref p2 xloc yloc p3 net p4 fanout]} {
;#Intermediate repeaters OK
                 lappend list_repeater $line
                 set lvlm1 [string map {"(" ""} $level]
                 set lvlm2 [string map {")" ""} $lvlm1]
                 set rep_level($count_repeater) $lvlm2
                 set rep_pin($count_repeater) $pin
                 set rep_cell($count_repeater) $cell_ref
                 set rep_xloc($count_repeater) [string map {"(" ""} $xloc]
                 set rep_yloc($count_repeater) [string map {")" ""} $yloc]
                 set rep_net($count_repeater) $net
                 set rep_fanout($count_repeater) $fanout
                 incr count_repeater
              }

              if {[regexp {^\s*(\S+)\s*(\S+)\s*(\S+)Ref: (\S+)]\s*(\S+)Location (\S+), (\S+)]\s*(\S+)Net:
(\S+)]\s*(\S+)ICG(\S+)\s*(\S+)Fanout: (\S+)]\s*$} $line match level pin p1 cell_ref p2 xloc yloc p3 net p4 p5
p6 fanout]} { ;#Intermediate ICGs OK
                 lappend list_ICG $line
                 set lvlm1 [string map {"(" ""} $level]
                 set lvlm2 [string map {")" ""} $lvlm1]
                 set ICG_level($count_ICG) $lvlm2
                 set ICG_pin($count_ICG) $pin
                 set ICG_cell($count_ICG) $cell_ref
                 set ICG_xloc($count_ICG) [string map {"(" ""} $xloc]
                 set ICG_yloc($count_ICG) [string map {")" ""} $yloc]
```

```
                set ICG_net($count_ICG) $net
                set ICG_fanout($count_ICG) $fanout
                incr count_ICG

            }

            if {[regexp {^\s*(\S+)\s*(\S+)\s*(\S+)Ref: (\S+)]\s*(\S+)Location (\S+), (\S+)]\s*(\S+)SINK
PIN(\S+)\s*$} $line match level pin p1 cell_ref p2 xloc yloc p3 p4]} { ;#Sink Pins OK
                lappend list_sink $line
                set lvlm1 [string map {"(" ""} $level]
                set lvlm2 [string map {")" ""} $lvlm1]
                set sink_level($count_sink) $lvlm2
                set sink_pin($count_sink) $pin
                set sink_cell($count_sink) $cell_ref
                set sink_xloc($count_sink) [string map {"(" ""} $xloc]
                set sink_yloc($count_sink) [string map {")" ""} $yloc]
                incr count_sink

            }

            if {[regexp {^\s*(\S+)\s*(\S+)\s*(\S+)Ref: (\S+)]\s*(\S+)Location (\S+), (\S+)]\s*(\S+)BALANCE
PIN(\S+)\s*$} $line match level pin p1 cell_ref p2 xloc yloc p3 p4]} { ;#Balance pins OK
                lappend list_balance $line
                set lvlm1 [string map {"(" ""} $level]
                set lvlm2 [string map {")" ""} $lvlm1]
                set balance_level($count_balance) $lvlm2
                set balance_pin($count_balance) $pin
                set balance_cell($count_balance) $cell_ref
                set balance_xloc($count_balance) [string map {"(" ""} $xloc]
                set balance_yloc($count_balance) [string map {")" ""} $yloc]
                incr count_balance

            }

        }
    }

    if {$debug_flag && $general_analysis_flag} {

        echo $count_source
        echo $count_repeater
        echo $count_ICG
        echo $count_sink
        echo $count_balance

        echo "rep level $rep_level(0)"
        echo "rep pin $rep_pin(0)"
        echo "rep cell $rep_cell(0)"
        echo "rep xloc $rep_xloc(0)"
        echo "rep yloc $rep_yloc(0)"
        echo "rep net $rep_net(0)"
        echo "rep fanout $rep_fanout(0)"

        echo "ICG level $ICG_level(0)"
        echo "ICG pin $ICG_pin(0)"
        echo "ICG cell$ICG_cell(0)"
        echo "ICG xloc $ICG_xloc(0)"
        echo "ICG yloc $ICG_yloc(0)"
        echo "ICG net $ICG_net(0)"
        echo "ICG fanout$ICG_fanout(0)"

        echo "sink level $sink_level(0)"
        echo "sink pin $sink_pin(0)"
        echo "sink cell $sink_cell(0)"
        echo "sink xloc $sink_xloc(0)"
        echo "sink yloc $sink_yloc(0)"
```

```
    echo "balance level $balance_level(0)"
    echo "balance pin $balance_pin(0)"
    echo "balance cell $balance_cell(0)"
    echo "balance xloc $balance_xloc(0)"
    echo "balance yloc $balance_yloc(0)"
}


if {$general_analysis_flag} {

    echo "\n"
    echo "Number of Source Pins: $count_source"
    echo "Number of Repeaters: $count_repeater"
    echo "Number of ICGs: $count_ICG"
    echo "Number of Sink Pins: $count_sink"
    echo "Number of Balance Pins: $count_balance"
    echo "\n"

    set count512 0
    set count256 0
    set count128 0
    set countl16 0
    set counth64 0
    set counth100 0

    foreach {repeater_fanout fanout} [array get rep_fanout] {
        if {$debug_flag} {
            echo "$repeater_fanout and $fanout"
        }
        if {$fanout >= $max_fanout_rep} {
            set max_fanout_rep $fanout
        }
        if {$fanout >= 32}  {incr count32}
        if {$fanout >= 128} {incr count128}
        if {$fanout >= 256} {incr count256}
        if {$fanout >= 512} {incr count512}
        if {$fanout < 16} {incr countl16}
        if {$fanout > 63} {incr counth64}
        if {$fanout > 100} {incr counth100}

        set average_fanout_rep [expr {$average_fanout_rep + $fanout}]
    }
    set average_fanout_rep [expr {$average_fanout_rep/$count_repeater}]

    echo "Max repeater fanout $max_fanout_rep"
    echo "Average repeater fanout $average_fanout_rep"
    echo "\n"

    foreach {ICG_fanout2 fanout} [array get ICG_fanout] {
        if {$debug_flag} {
            echo "$ICG_fanout2 and $fanout"
        }
        if {$fanout >= $max_fanout_ICG} {
            set max_fanout_ICG $fanout
        }

        if {$fanout >= 32}  {incr count32}
        if {$fanout >= 128} {incr count128}
        if {$fanout >= 256} {incr count256}
        if {$fanout >= 512} {incr count512}
        if {$fanout < 16} {incr countl16}
        if {$fanout > 63} {incr counth64}
        if {$fanout > 100} {incr counth100}

        set average_fanout_ICG [expr {$average_fanout_ICG + $fanout}]
```

```
        }
        set average_fanout_ICG [expr {$average_fanout_ICG/$count_ICG}]

        echo "Max ICG fanout $max_fanout_ICG"
        echo "Average ICG fanout $average_fanout_ICG"
        #echo "\n"

        echo "\n"
        echo "General fanout breakdown information"
        echo "Number of drivers with fanout equal or bigger than 512: $count512"
        echo "Number of drivers with fanout equal or bigger than 256: $count256"
        echo "Number of drivers with fanout equal or bigger than 128: $count128"
        echo "Number of drivers with fanout equal or bigger than 32: $count32"
        echo "Number of drivers with fanout higher than 100: $counth100"
        echo "Number of drivers with fanout higher or equal than 64: $counth64"
        echo "Number of drivers with fanout lower than 16: $countl16"


        if {$cells_flag} {
           echo "Cells breakdown information\n"

set ccount_rep 0

### Obtention of clock cells via list
### Repeater analysis:

foreach {cell_number cell} [array get rep_cell] {
   lappend repeater_list $cell
   set repeater_unique [lsort -unique $repeater_list]
}
foreach clockcell $repeater_unique {
   if {$clockcell == [lindex $repeater_unique $ccount_rep]} {
      set cell_type_repeater($ccount_rep) $clockcell
      set cell_rep1($ccount_rep) [lsearch -all $repeater_list $clockcell]
      set length_cell_rep1($ccount_rep) [llength $cell_rep1($ccount_rep)]
      if {$length_cell_rep1($ccount_rep) == 0} {break}
      incr ccount_rep
   }
}
#echo "Repeater cell breakdown\n"
if {$ccount_rep == 1} {
   echo "Cell $cell_type_repeater(0) Number of instances of $length_cell_rep1(0)"
} else {
   for {set counter 0} {$counter < $ccount_rep } {incr counter} {
      echo "Cell $cell_type_repeater($counter) Number of instances $length_cell_rep1($counter)"
   }
}

            ### ICG analysis:
            set ccount_ICG 0

            foreach {ICG_number ICG} [array get ICG_cell] {
               lappend ICG_list $ICG
               set ICG_unique [lsort -unique $ICG_list]
            }
            foreach ICGcell $ICG_unique {
               if {$ICGcell == [lindex $ICG_unique $ccount_ICG]} {
                  set cell_type_ICG($ccount_ICG) $ICGcell
                  set cell_ICG1($ccount_ICG) [lsearch -all $ICG_list $ICGcell]
                  set length_cell_ICG1($ccount_ICG) [llength $cell_ICG1($ccount_ICG)]
                  if {$length_cell_ICG1($ccount_ICG) == 0 } {break}
                  incr ccount_ICG
               }
            }
            echo "ICG cells breakdown\n"
            if {$ccount_ICG == 1} {
```

```
            echo "Cell $cell_type_ICG(0) Number of instances $length_cell_ICG1(0)"
         } else {
            for {set counterICG 0} {$counterICG < $ccount_ICG} {incr counterICG} {
               echo "Cell $cell_type_ICG($counterICG) Number of instances
$length_cell_ICG1($counterICG)"
            }
         }
         #echo "\n"

         ### Leaf cell analysis:
         set ccount_sink 0

         foreach {leaf_number leaf} [array get sink_cell] {
            lappend leaf_list $leaf
            set leaf_unique [lsort -unique $leaf_list]
         }

         foreach leafcell $leaf_unique {
            if {$leafcell == [lindex $leaf_unique $ccount_sink]} {
               set cell_type_leaf($ccount_sink) $leafcell
               set cell_leaf1($ccount_sink) [lsearch -all $leaf_list $leafcell]
               set length_cell_leaf1($ccount_sink) [llength $cell_leaf1($ccount_sink)]
               if {$length_cell_leaf1($ccount_sink) == 0 } {break}
               incr ccount_sink

            }
         }
         echo "Leaf cells breakdown\n"
         if {$ccount_sink == 1} {
            echo "Cell $cell_type_leaf(0) Number of instances $length_cell_leaf1(0)"
         } else {
            for {set counterleaf 0} {$counterleaf < $ccount_sink} {incr counterleaf} {
               echo "Cell $cell_type_leaf($counterleaf) Number of instances
$length_cell_leaf1($counterleaf)"
            }
         }
         #echo "\n"

         ### Balance cell analysis:
         set ccount_balance 0

         foreach {leaf_number balance} [array get balance_cell] {
            lappend balance_list $balance
            set balance_unique [lsort -unique $balance_list]
         }

         foreach balancecell $balance_unique {
            if {$balancecell == [lindex $balance_unique $ccount_balance]} {
               set cell_type_balance($ccount_balance) $balancecell
               set cell_balance1($ccount_balance) [lsearch -all $balance_list $balancecell]
               set length_cell_balance1($ccount_balance) [llength $cell_balance1($ccount_balance)]
               if {$length_cell_balance1($ccount_balance) == 0 } {break}
               incr ccount_balance

            }
         }
         echo "Balance cells breakdown\n"
         if {$ccount_balance == 1} {
            echo "Cell $cell_type_balance(0) Number of instances $length_cell_balance1(0)"
         } else {
            for {set counterbalance 0} {$counterbalance < $ccount_balance} {incr counterbalance} {
               echo "Cell $cell_type_balance($counterbalance) Number of instances
$length_cell_balance1($counterbalance)"
            }
         }
```

```
        }

    }

    #echo "\n"
    ### Made with elseif to avoid use of flags to go to next line of loop
    if {$structure_flag} {
        #If the structure flag exists it will use either the selected (if existing) or the existing structure generated by the reporter.
        if {[file exists "reports/routeopt/selected/clock_QoR_structure.rpt"]} {

            set text_file [open "reports/routeopt/selected/clock_QoR_structure.rpt" "r"]
        } else {

            set text_file [open "reports/routeopt/clock/clock_QoR_structure.rpt" "r"]
        }

        set x [read $text_file]

    } else {
        #If structure flag isnt set, it generates the report and saves it on selected directory
        redirect {*}$tee x {report_clock_QoR -type structure -nosplit}

        echo "$x" > "reports/routeopt/selected/clock_QoR_structure.rpt"
        echo "New structure is is saved on selected directory preserving the existing one\n"
    }



    if {$endpoint_analysis_flag} {
        set sink_number2 0
        set balance_number2 0


        foreach line [split $x "\n"] {
            #echo $line_pointer
            if {[regexp {^\s*(\S+)\s*(\S+)\s*(\S+)in Port(\S+)\s*(\S+)Location (\S+), (\S+)]\s*(\S+)Net: (\S+)]\s*(\S+)Fanout: (\S+)]\s*$} $line match level pin p1 p2 p3 xloc yloc p4 net p5 fanout]} { ;#Level zero OK
                echo "First line is not considered as it always drives a repeater/ICG on single source"
                echo "When source is met, counter starts"
                set lvl1_a [string map {"(" ""} $level]
                set lvl2_a [string map {")" ""} $lvl1_a]
                set level_a($line_pointer) $lvl2_a

                lappend list_celltype "source"
                lappend list_fanout $fanout
                lappend list_level $lvl2_a
                lappend list_xloc [string map {"(" ""} $xloc]
                lappend list_yloc [string map {")" ""} $yloc]
                lappend list_pointer $line_pointer
                lappend list_pin $pin
                incr line_pointer
            } elseif {[regexp {^\s*(\S+)\s*(\S+)\s*(\S+)Ref: (\S+)]\s*(\S+)Location (\S+), (\S+)]\s*(\S+)Net: (\S+)]\s*(\S+)Fanout: (\S+)]\s*$} $line match level pin p1 cell_ref p2 xloc yloc p3 net p4 fanout]} {
;#Intermediate repeaters
                set lvl1_a [string map {"(" ""} $level]
                set lvl2_a [string map {")" ""} $lvl1_a]
                set level_a($line_pointer) $lvl2_a
                lappend list_pointer $line_pointer
                lappend list_celltype "driver"
                lappend list_fanout $fanout
                lappend list_level $lvl2_a
                lappend list_xloc [string map {"(" ""} $xloc]
                lappend list_yloc [string map {")" ""} $yloc]
                lappend list_pin $pin
```

```
                incr line_pointer
            } elseif {[regexp {^\s*(\S+)\s*(\S+)\s*(\S+)Ref: (\S+)]\s*(\S+)Location (\S+), (\S+)]\s*(\S+)Net:
(\S+)]\s*(\S+)ICG(\S+)\s*(\S+)Fanout: (\S+)]\s*$} $line match level pin p1 cell_ref p2 xloc yloc p3 net p4 p5
p6 fanout]} { ;#Intermediate ICGs OK
                set lvl1_a [string map {"(" ""} $level]
                set lvl2_a [string map {")" ""} $lvl1_a]
                set level_a($line_pointer) $lvl2_a
                lappend list_pointer $line_pointer
                lappend list_celltype "driver"
                lappend list_fanout $fanout
                lappend list_level $lvl2_a
                lappend list_xloc [string map {"(" ""} $xloc]
                lappend list_yloc [string map {")" ""} $yloc]
                lappend list_pin $pin

                incr line_pointer
            } elseif {[regexp {^\s*(\S+)\s*(\S+)\s*(\S+)Ref: (\S+)]\s*(\S+)Location (\S+), (\S+)]\s*(\S+)SINK
PIN(\S+)\s*$} $line match level pin p1 cell_ref p2 xloc yloc p3 p4]} { ;#Sink Pins OK
                set lvl1_a [string map {"(" ""} $level]
                set lvl2_a [string map {")" ""} $lvl1_a]
                set level_a($line_pointer) $lvl2_a
                lappend list_pointer $line_pointer
                lappend list_celltype "sink"
                lappend list_fanout "sink"
                lappend list_level $lvl2_a
                lappend list_xloc [string map {"(" ""} $xloc]
                lappend list_yloc [string map {")" ""} $yloc]
                lappend list_pin $pin

                incr line_pointer
                incr sink_number2
            } elseif {[regexp {^\s*(\S+)\s*(\S+)\s*(\S+)Ref: (\S+)]\s*(\S+)Location (\S+),
(\S+)]\s*(\S+)BALANCE PIN(\S+)\s*$} $line match level pin p1 cell_ref p2 xloc yloc p3 p4]} { ;#Balance pins
OK
                set lvl1_a [string map {"(" ""} $level]
                set lvl2_a [string map {")" ""} $lvl1_a]
                set level_a($line_pointer) $lvl2_a
                lappend list_pointer $line_pointer
                lappend list_celltype "sink"
                lappend list_fanout "sink"
                lappend list_level $lvl2_a
                lappend list_xloc [string map {"(" ""} $xloc]
                lappend list_yloc [string map {")" ""} $yloc]
                lappend list_pin $pin

                incr line_pointer
              incr balance_number2
              }

          }
          echo $sink_number2
          echo $balance_number2
          set line_pointer [expr {$line_pointer - 1}] ;#Last increase leads to void data
          ### Variable declaration:

          set driver 0
          set endpoint 0
          set sep "_"


        set driver 0
        set endpoint 0
        set sep "_"

        set current_xloc 0
        set current_yloc 0
```

```
set current_fanout 0

set line_pointer_max [expr {$line_pointer + 1}]
set unique_fanout 0

foreach value_pointer $list_pointer {
   set next_pointer [expr {$value_pointer + 1}]
   set prev_pointer [expr {$value_pointer - 1}]

   set current_celltype [lindex $list_celltype $value_pointer]
   set current_fanout [lindex $list_fanout $value_pointer]
   set current_xloc [lindex $list_xloc $value_pointer]
   set current_yloc [lindex $list_yloc $value_pointer]
   set current_level [lindex $list_level $value_pointer]
   set current_pin [lindex $list_pin $value_pointer]

   set next_celltype [lindex $list_celltype $next_pointer]
   set next_level [lindex $list_level $next_pointer]
   #echo "$current_fanout $current_celltype"

   if {$current_celltype == "driver" && $current_fanout == 1} {
      if {$next_celltype == "driver"} {
         echo "Intermediate driver - Ignored on endpoint driver considerations"
      } elseif {$next_celltype == "sink"} {
         incr unique_fanout
         set repICG_driver_xloc($driver) $current_xloc
         set repICG_driver_yloc($driver) $current_yloc
         set repICG_driver_fanout($driver) [expr {$current_fanout - 1}]
         set repICG_driver_pin($driver) $current_pin

         set repICG_id($driver) $driver$current_celltype$current_fanout

         set endpoint 0
         set repICG_endpoint_xloc($driver$sep$endpoint) [lindex $list_xloc $next_pointer]
         set repICG_endpoint_yloc($driver$sep$endpoint) [lindex $list_yloc $next_pointer]
         set repICG_endpoint_celltype($driver$sep$endpoint) "sink"
         set repICG_endpoint_pin($driver$sep$endpoint) [lindex $list_pin $next_pointer]

         set repICG_endpoint_pin($driver$sep$endpoint) [lindex $list_pin $next_pointer]

         set endpoint_id($driver$sep$endpoint) $driver$next_celltype$endpoint$sep$current_level
         #echo $driver
         incr driver
         set aux_driver [expr {$driver -1 }]

         set repICG_driver_endpoint($aux_driver) 0


      }
   }
   if {$current_celltype == "driver" && $current_fanout > 1} {
      set repICG_driver_xloc($driver) $current_xloc
      set repICG_driver_yloc($driver) $current_yloc
      set repICG_driver_pin($driver) $current_pin
      set repICG_driver_fanout($driver) [expr {$current_fanout - 1}]
      #set repICG_driver_celltype($driver) $current_celltype

      set repICG_id($driver) $driver$current_celltype$current_fanout

      set endpoint 0
      incr driver

      set driver_endpoint [expr {$driver - 1}]
      set level_sinks [expr {$current_level + 1}]
      set new_pointer [expr {$value_pointer - 1}]
```

```
            for {set endpoint_fanout $value_pointer} {$endpoint_fanout < $line_pointer_max} {incr
endpoint_fanout} {; #All next value pointers.

                set aux_level [lindex $list_level $endpoint_fanout]
                if {$aux_level == $level_sinks} {
                    set sink_celltype [lindex $list_celltype $endpoint_fanout]
                    set repICG_endpoint_xloc($driver_endpoint$sep$endpoint) [lindex $list_xloc
$endpoint_fanout]
                    set repICG_endpoint_yloc($driver_endpoint$sep$endpoint) [lindex $list_yloc
$endpoint_fanout]
                    set repICG_endpoint_celltype($driver_endpoint$sep$endpoint) [lindex $list_celltype
$endpoint_fanout]
                    set repICG_endpoint_pin($driver_endpoint$sep$endpoint) [lindex $list_pin $endpoint_fanout]

                    set endpoint_id($driver_endpoint$sep$endpoint)
$driver_endpoint$sink_celltype$endpoint$sep$aux_level

                    incr endpoint

                    if {$endpoint == $current_fanout} {
                        if {$driver == 172} {echo "in"}
                        set aux_driver [expr {$driver -1 }]

                        set repICG_driver_endpoint($aux_driver) [expr {$endpoint - 1}]

                        incr driver_endpoint

                        break
                    }
                }
            }

            set aux_driver_endpoint [expr {$driver_endpoint - 1}]

            for {set noendpoint 0} {$noendpoint < $endpoint} {incr noendpoint} {
                set aux_endpoint [expr {$endpoint - 1}]
                if {$repICG_endpoint_celltype($aux_driver_endpoint$sep$noendpoint) == "sink"} {
                    #echo "sink found"
                    break
                } elseif {$noendpoint == $aux_endpoint} {
                    set driver [expr {$driver - 1}]
                    set driver_endpoint [expr {$driver_endpoint - 1}]
                    #echo "All drivers!"
                }
            }


        }
        #echo $driver
        if {$value_pointer == $line_pointer_max} {break} ;#$line_pointer_max
    }

            foreach {endpoint_number endpoint_fanout_array} [array get repICG_driver_endpoint] {
                #echo "$endpoint_number for $endpoint_fanout_array"
                lappend end_list $endpoint_fanout_array
                set length_lst [llength $end_list]

                #set leaf_unique [lsort -unique $leaf_list]
            }
echo $length_lst

    set effective_driver [expr {$driver - 1}]
    set csv_file "reports/routeopt/selected/ordering.csv"
    set csv [open $csv_file "w"]
    for {set counter_driver 0} {$counter_driver < $driver} {incr counter_driver} {
```

```
        puts $csv "$rep/CG_id($counter_driver)"
        for {set counter_endpoint 0} {$counter_endpoint <= $rep/CG_driver_endpoint($counter_driver)} {incr
counter_endpoint} {
            puts $csv "\t$endpoint_id($counter_driver$sep$counter_endpoint)"
        }
    }
    #echo $driver
    close $csv
    ### Sink counter to see if math checks
    if {$debug_flag && $endpoint_analysis_flag} {

        set max_endpoint([expr {$driver - 1}]) $endpoint
        set tot_endpoints 0

        for {set counter_driver 0} {$counter_driver < $driver} {incr counter_driver} {
            #echo $max_endpoint($counter_driver)
            set tot_endpoints [expr {$tot_endpoints + $max_endpoint($counter_driver)}]
        }
        #echo $tot_endpoints

        foreach {id_number identifier} [array get rep/CG_id] {
            lappend array_ids $identifier
            set array_ordered [lsort -dictionary $array_ids]
            #echo "Array position $id_number with identifier $identifier"

        }
        #foreach array_pos $array_ordered {echo $array_pos}
    }

    ##### MANHATTAN DISTANCE #####

    set max_manhattan_xdistance 0
    set max_manhattan_ydistance 0
    set max_manhattan_distance 0

    set total_manhattan_xdistance 0
    set total_manhattan_ydistance 0
    set total_manhattan_distance 0

    echo "\n"

    set csv_file "reports/routeopt/selected/manhattan_test.csv"
    set csv [open $csv_file "w"]

    ### Note that all manhattan distance calculations are done for any endpoint driver to all its fanout.
Intermediate drivers on the hybrid fanout are also taken into consideration.

    for {set counter_driver 0} {$counter_driver < $driver} {incr counter_driver} {
        set temporal_driver_xloc $rep/CG_driver_xloc($counter_driver)
        set temporal_driver_yloc $rep/CG_driver_yloc($counter_driver)

        puts $csv "$temporal_driver_xloc, $temporal_driver_yloc, $rep/CG_id($counter_driver),
$rep/CG_driver_fanout($counter_driver)"

        set max_manhattan_xdistance_driver($counter_driver) 0
        set max_manhattan_ydistance_driver($counter_driver) 0

        set max_manhattan_distance_driver($counter_driver) 0

        set total_manhattan_xdistance_driver($counter_driver) 0
        set total_manhattan_ydistance_driver($counter_driver) 0

        set total_manhattan_distance_driver($counter_driver) 0

        for {set counter_endpoint 0} {$counter_endpoint <= $rep/CG_driver_endpoint($counter_driver)} {incr
counter_endpoint} {
```

```
            set temporal_endpoint_xloc $rep/CG_endpoint_xloc($counter_driver$sep$counter_endpoint)
            set temporal_endpoint_yloc $rep/CG_endpoint_yloc($counter_driver$sep$counter_endpoint)

            set manhattan_xdistance_temporal [expr {$temporal_endpoint_xloc - $temporal_driver_xloc}]
            set manhattan_ydistance_temporal [expr {$temporal_endpoint_yloc - $temporal_driver_yloc}]

            set manhattan_xdistance($counter_driver$sep$counter_endpoint) [expr
{abs($manhattan_xdistance_temporal)}]
            set manhattan_ydistance($counter_driver$sep$counter_endpoint) [expr
{abs($manhattan_ydistance_temporal)}]

            set manhattan_distance($counter_driver$sep$counter_endpoint) [expr
{$manhattan_xdistance($counter_driver$sep$counter_endpoint) +
$manhattan_ydistance($counter_driver$sep$counter_endpoint)}]

            puts $csv "\t$temporal_endpoint_xloc, $temporal_endpoint_yloc,
$endpoint_id($counter_driver$sep$counter_endpoint),
$manhattan_distance($counter_driver$sep$counter_endpoint)"

            ### Max overall manhattan distance

            if {$manhattan_xdistance($counter_driver$sep$counter_endpoint) > $max_manhattan_xdistance}
{set max_manhattan_xdistance $manhattan_xdistance($counter_driver$sep$counter_endpoint)}
            if {$manhattan_ydistance($counter_driver$sep$counter_endpoint) > $max_manhattan_ydistance}
{set max_manhattan_ydistance $manhattan_ydistance($counter_driver$sep$counter_endpoint)}

            ### Max manhattan distance for each individual driver

            if {$manhattan_xdistance($counter_driver$sep$counter_endpoint) >
$max_manhattan_xdistance_driver($counter_driver)} {set
max_manhattan_xdistance_driver($counter_driver)
$manhattan_xdistance($counter_driver$sep$counter_endpoint)}
            if {$manhattan_ydistance($counter_driver$sep$counter_endpoint) >
$max_manhattan_ydistance_driver($counter_driver)} {set
max_manhattan_ydistance_driver($counter_driver)
$manhattan_ydistance($counter_driver$sep$counter_endpoint)}
            if {$manhattan_distance($counter_driver$sep$counter_endpoint) >
$max_manhattan_distance_driver($counter_driver)} {set max_manhattan_distance_driver($counter_driver)
$manhattan_distance($counter_driver$sep$counter_endpoint)}

            ### Average manhattan distance overall

            set total_manhattan_xdistance [expr {$total_manhattan_xdistance +
$manhattan_xdistance($counter_driver$sep$counter_endpoint)}]
            set total_manhattan_ydistance [expr {$total_manhattan_ydistance +
$manhattan_ydistance($counter_driver$sep$counter_endpoint)}]
            set total_manhattan_distance [expr {$total_manhattan_distance +
$manhattan_distance($counter_driver$sep$counter_endpoint)}]

            ### Average manhattan distance for each individual driver

            set total_manhattan_xdistance_driver($counter_driver) [expr
{$total_manhattan_xdistance_driver($counter_driver) +
$manhattan_xdistance($counter_driver$sep$counter_endpoint)}]
            set total_manhattan_ydistance_driver($counter_driver) [expr
{$total_manhattan_ydistance_driver($counter_driver) +
$manhattan_ydistance($counter_driver$sep$counter_endpoint)}]
            set total_manhattan_distance_driver($counter_driver) [expr
{$total_manhattan_distance_driver($counter_driver) +
$manhattan_distance($counter_driver$sep$counter_endpoint)}]

        }
    }
    close $csv
```

```
      for {set counter_driver 0} {$counter_driver < $driver} {incr counter_driver} {
         set effective_endpoint [expr {$rep/CG_driver_fanout($counter_driver) + 1}]

         set average_xdistance_driver($counter_driver) [expr
{$total_manhattan_xdistance_driver($counter_driver)/$effective_endpoint}]
         set average_ydistance_driver($counter_driver) [expr
{$total_manhattan_ydistance_driver($counter_driver)/$effective_endpoint}]
         set average_distance_driver($counter_driver) [expr {$average_xdistance_driver($counter_driver) +
$average_ydistance_driver($counter_driver)}]
      }
      }

      ##

      if {$endpoint_analysis_flag && $capacitance_analysis_flag} {
         ### Gathering net fanout based on endpoint drivers
         ### Conversion from pins to nets using get_attribute
         for {set counter_driver 0} {$counter_driver < $driver} {incr counter_driver} {
            for {set counter_endpoint 0} {$counter_endpoint <= $rep/CG_driver_endpoint($counter_driver)}
{incr counter_endpoint} {
               set temporal_celltype_value
$rep/CG_endpoint_celltype($counter_driver$sep$counter_endpoint)
               if {$temporal_celltype_value == "sink"} {
                  set temporal_net [get_attribute -class pin -name net_name -objects
$rep/CG_endpoint_pin($counter_driver$sep$counter_endpoint)]
               } else {
                  set temporal_net "nil"
               }
               ### Populates intermediate list with all values from temporal net.
               if {$temporal_net != "nil"} {
                  lappend temporal_list $temporal_net
               }

            }
            ### When exiting first loop, sorts all unique values from existing list and stores the list into an array
            set net_list_driver [lsort -unique $temporal_list]
            set net_list($counter_driver) $net_list_driver


            set temporal_list {}


         }
         for {set counter_driver 0} {$counter_driver < $driver} {incr counter_driver} {
            set total_wire_cap($counter_driver) 0
            set total_total_cap($counter_driver) 0
            set total_pin_cap($counter_driver) 0
            foreach value $net_list($counter_driver) {


               ### Extraction of the individual values
               set temp_wire_cap [get_attribute -class net -name wire_capacitance_max -objects $value]
               set temp_total_cap [get_attribute -class net -name total_capacitance_max -objects $value]


               if {$temp_wire_cap != "" && $temp_total_cap != ""} {
                  set temp_pin_cap [expr {$temp_total_cap - $temp_wire_cap}]
               } else {
                  set temp_wire_cap 0
                  set temp_total_cap 0
                  set temp_pin_cap 0
               }
```

```
            set total_wire_cap($counter_driver) [expr {$total_wire_cap($counter_driver) +
$temp_wire_cap}]
            set total_total_cap($counter_driver) [expr {$total_total_cap($counter_driver) +
$temp_total_cap}]
            set total_pin_cap($counter_driver) [expr {$total_pin_cap($counter_driver) + $temp_pin_cap}]

            lappend wire_cap($counter_driver) $temp_wire_cap
            lappend total_cap($counter_driver) $temp_total_cap
            lappend pin_cap($counter_driver) $temp_pin_cap

      if {$csv_flag} {
         echo ".csv file generation\n"
         set csv_file "reports/routeopt/selected/QoR_clock_structure.csv"
         set csv [open $csv_file "w"]

         puts $csv "sep=,"

         if {$general_analysis_flag} {
            echo "General data formatting\n"
            puts $csv "Number of source pins, Number of repeaters, Number of ICGs, Number of sink
pins, Number of balance pins"
            puts $csv "$count_source, $count_repeater, $count_ICG, $count_sink, $count_balance"
            puts $csv "Maximum repeater fanout, Average repeater fanout, Maximum ICG fanout,
Average ICG fanout"
            puts $csv "$max_fanout_rep, $average_fanout_rep, $max_fanout_ICG,
$average_fanout_ICG"

            puts $csv "\n"
            puts $csv "General fanout breakdown information"
            puts $csv "#Drivers fanout >= 512, $count512"
            puts $csv "#Drivers fanout >= 256, $count256"
            puts $csv "#Drivers fanout >= 128, $count128"
            puts $csv "#Drivers fanout >= 32, $count32"
            puts $csv "#Drivers fanout > 100, $counth100"
            puts $csv "#Drivers fanout >= 64, $counth64"
            puts $csv "#Drivers fanout < 16, $countl16"

            if {$ccount_rep == 1} {
               puts $csv "\n"
               puts $csv "Repeater cells breakdown\n"
               puts $csv "Cell reference, Number of instances"
               puts $csv "$cell_type_repeater(0), $length_cell_rep1(0)"
            } else {
               puts $csv "\n"
               puts $csv "Repeater cells breakdown\n"
               puts $csv "Cell reference, Number of instances"
               for {set counter 0} {$counter < $ccount_rep} {incr counter} {
                  puts $csv "$cell_type_repeater($counter), $length_cell_rep1($counter)"
               }
            }
            if {$ccount_ICG == 1} {
               puts $csv "\n"
               puts $csv "ICG cells breakdown\n"
               puts $csv "Cell reference, Number of instances"
               puts $csv "$cell_type_ICG(0), $length_cell_ICG1(0)"
            } else {
               puts $csv "\n"
               puts $csv "Repeater cells breakdown\n"
               puts $csv "Cell reference, Number of instances"
               for {set counter 0} {$counter < $ccount_ICG} {incr counter} {
                  puts $csv "$cell_type_ICG($counter), $length_cell_rep1($counter)"
               }
            }
            if {$ccount_sink == 1} {
               puts $csv "\n"
```

```
            puts $csv "Sink cells breakdown\n"
            puts $csv "Cell reference, Number of instances"
            puts $csv "$cell_type_leaf(0), $length_cell_leaf1(0)"
         } else {
            puts $csv "\n"
            puts $csv "Sinks cells breakdown\n"
            puts $csv "Cell reference, Number of instances"
            for {set counter 0} {$counter < $ccount_sink} {incr counter} {
               puts $csv "$cell_type_leaf($counter), $length_cell_leaf1($counter)"
            }
         }

         if {$ccount_balance} {
            puts $csv "\n"
            puts $csv "Balance cells breakdown\n"
            puts $csv "Cell reference, Number of instances"
            puts $csv "$cell_type_balance(0), $length_cell_balance1(0)"
         } else {
            puts $csv "\n"
            puts $csv "Balance cells breakdown\n"
            puts $csv "Cell reference, Number of instances"
            for {set counter 0} {$counter < $ccount_balance} {incr counter} {
               puts $csv "$cell_type_balance($counter), $length_cell_balance1($counter)"
            }
         }
      }
      if {$endpoint_analysis_flag && $capacitance_analysis_flag} {
         puts $csv "\n"
         puts $csv "Driver capacitance and Manhattan distance analysis"
         puts $csv "Downstream capacitance information, on endpoint drivers only sink and balance
capacitance will be considered.\n"
         puts $csv "Driver Idenfifier, Driver fanout, Max x-distance, Max y-distance, Max Distance,
Average x-distance, Average y-distance, Average distance, Driver Total Capacitance, Driver Pin
capacitance, Driver Wire Capacitance"
         for {set counter_driver 0} {$counter_driver < $driver} {incr counter_driver} {

            puts $csv "$repICG_id($counter_driver), $repICG_driver_fanout($counter_driver),
$max_manhattan_xdistance_driver($counter_driver),
$max_manhattan_ydistance_driver($counter_driver),
$max_manhattan_distance_driver($counter_driver), $average_xdistance_driver($counter_driver),
$average_ydistance_driver($counter_driver), $average_distance_driver($counter_driver),
$total_total_cap($counter_driver), $total_pin_cap($counter_driver), $total_wire_cap($counter_driver)"
         }
         if {$detailed_csv_flag} {
            echo "Driver fanout and manhattan distances\n"
            puts $csv "Driver identifier, Driver fanout, Driver x-location, Driver y-location"
            for {set counter_driver 0} {$counter_driver < $driver} {incr counter_driver} {
               puts $csv "$repICG_id($counter_driver), $repICG_driver_fanout($counter_driver),
$repICG_driver_xloc($counter_driver), $repICG_driver_yloc($counter_driver)"
               puts $csv "\tEndpoint identifier, Endpoint x-location, Endpoint y-location, Driver
manhattan distance"
               for {set counter_endpoint 0} {$counter_endpoint <=
$repICG_driver_endpoint($counter_driver)} {incr counter_endpoint} {
                  puts $csv "\t$endpoint_id($counter_driver$sep$counter_endpoint),
$repICG_endpoint_xloc($counter_driver$sep$counter_endpoint),
$repICG_endpoint_yloc($counter_driver$sep$counter_endpoint),
$manhattan_distance($counter_driver$sep$counter_endpoint)"
               }
            }
         }
      }

      close $csv
   }
  }
}
```

```
define_proc_attributes clock_structure_analyzer -info "Extracts information from the clock structure" \
   -define_args {
      {-tee "Optional - displays the output of under-the-hood clock_structure_analyzer command" "" boolean optional}
      {-structure "Optional - specifies if existing structure is used or new file is generated via report_clock_QoR -type  structure" "" boolean optional}
      {-debugger "Optional - Sets up a flag to see debugging information" "" boolean optional}
      {-repeater_cells "Optional - Specifies if cells analisis is done" "" boolean optional}
      {-general_analysis "Optional - Specifies if general analysis on cell distribution and other metrics is done" "" boolean optional}
      {-endpoint_analysis "Optional - Specifies if analysis on the sinks/balance and their drivers is done" "" boolean optional}
      {-csv_file "Optional - specifies that a .csv file will be generated storing available info of other options" "" boolean optional}
      {-detailed_csv "Optional - Adds detailed information on manhattan distance for each endpoint and all endpoint drivers" "" boolean optional}
      {-capacitance_analysis "Optional - Adds information on net fanout" "" boolean optional}
   }

echo "\tclock_structure_analyzer"
```

Appendix 21: Script *replace_clockpin.tcl*

```
### Script to reposition the input clock pin:

proc replace_clockpin {args} {
   parse_proc_arguments -args $args results

   set shape_type $results(-shape)
   set init_point $results(-init_point)
   set end_point $results(-end_point)
   set clock_name $results(-clock_name)
   set bbox_margin $results(-bbox_margin)
   set layer $results(-layer)

   set i 0
   foreach point $init_point {
      set initial_point($i) $point
      #echo $initial_point($i)
      incr i
   }

   set i 0

   foreach point $end_point {
      set final_point($i) $point
      #echo $final_point($i)
      incr i
   }
   for {set j 0} {$j < 2} {incr j} {
      set midpoint($j) [expr {$initial_point($j) + $final_point($j)}]
      set averagepoint($j) [expr {$midpoint($j)/2}]
      echo $averagepoint($j)
   }
   set bbox_minus_x [expr {$averagepoint(0) - $bbox_margin}]
   set bbox_minus_y [expr {$averagepoint(1) - $bbox_margin}]

   set bbox_plus_x [expr {$averagepoint(0) + $bbox_margin}]
   set bbox_plus_y [expr {$averagepoint(1) + $bbox_margin}]

   echo "{$bbox_minus_x $bbox_minus_y}{$bbox_plus_x $bbox_plus_y}"
```

```
   set sysClk_shape [get_shapes -of_objects [get_ports -of_objects $clock_name]]
   remove_shapes -force -verbose $sysClk_shape
   echo "create_shape -shape_type $shape_type -layer $layer -shape_use macro_pin_connect -port
$clock_name -boundary {{$bbox_minus_x $bbox_minus_y} {$bbox_plus_x $bbox_plus_y}}"
   set aux_boundary "{{$bbox_minus_x $bbox_minus_y} {$bbox_plus_x $bbox_plus_y}}"
   create_shape -shape_type $shape_type -layer $layer -shape_use macro_pin_connect -port $clock_name -
boundary $aux_boundary



}

define_proc_attributes replace_clockpin -info "Creates a new input clock point" \
   -define_args {
         {-tee "Optional - displays the output under-the-hood replace_clockpin command" "" boolean
optional}
         {-init_point "Optional - x,y  base point collection of the block" "" list optional}
         {-end_point "Optional - x,y final point collection of the block" "" list optional}
         {-bbox_margin "Optional - margin left on the bbox from the center point to the extremes" "" float
optional}
         {-clock_name "Optional - name of the clock port to be moved" "" string optional}
         {-shape "Optional - boundary type" "" string optional}
         {-layer "Optional - layer to be moved " "" string optional}

   }
echo "\treplace_clockpin"
```

Appendix 22: Script *block_combiner.tcl*

```
proc block_combiner {args} {
   parse_proc_arguments -args $args results

   ### Flag definition

   set block_4_exists     [info exists results(-block_4_list)]
   set block_1_exists          [info exists results(-block_1_list)]
   set block_2_exists          [info exists results(-block_2_list)]
   set multi_files_flag      [info exists results(-separate_files)]
   set math_flag          [info exists results(-math)]
   set block_3_exists          [info exists results(-block_3_list)]

   ### Pre_lists populating

   if {$block_4_exists} {
      set list_block_4 $results(-block_4_list)
   }
   if {$block_1_exists} {
      set list_block_1 $results(-block_1_list)
   }
   if {$block_2_exists} {
```

```
      set list_block_2 $results(-block_2_list)
   }
   if {$block_3_exists} {
      set list_block_3 $results(-block_3_list)
   }

   ### -tee settings:
   if {[info exists results(-tee)]} {
      set tee "-tee -var"
   } else {
      set tee "-var"
   }


   ### List generation  for metrics:

   list clock_QoR_block_4 ""
   list clock_QoR_Block_1 ""
   list clock_QoR_Block_2 ""
   list clock_QoR_block_3 ""

   list general_block_4 ""
   list general_Block_1 ""
   list general_Block_2 ""
   list general_block_3 ""

   set clock_QoR_flag 0
   set general_QoR_flag 0
   set iteration_flag 0

list blocklist ""
if {$block_4_exists} {lappend blocklist "block_4"}
if {$Block_1_exists} {lappend blocklist "block_1"}
if {$Block_2_exists} {lappend blocklist "block_2"}
if {$block_3_exists} {lappend blocklist "block_3"}

   #set blocklist {block_4 Block_1 Block_2}

   set clock_QoR_baseline "Set, WNS (ns), TNS (ns), Setup FEP, Clocks*lew* FEP, *Skew* (ns), *Latency* (ns), Clock Cells, Clock *Repeaters*,  Area Clock Cells (um2), *Repeater* Area (um2), Dynamic Clock Power (uW), Leakage Clock Power (uW), Total Clock Power (uW)"
   set general_QoR_baseline "Set, Utilization, Total Area (um2), Total Stdcell Area (um2), Total Dynamic Power (uW), Total Leakage Power (uW), Total Power (uW), Number of *DRC*s"


      echo "A .csv file will be generated by each existing block"

      ## Index a foreach with each list name
      list temporal_list {}
      foreach blocktype $blocklist {
         if {$blocktype == "block_4" && $block_4_exists} {set temporal_list $list_block_4}
         if {$blocktype == "block_1" && $block_1_exists} {set temporal_list $list_block_1}
         if {$blocktype == "block_2" && $block_2_exists} {set temporal_list $list_block_2}
         if {$blocktype == "block_3" && $block_3_exists} {set temporal_list $list_block_3}

         foreach blockname $temporal_list {
            #set temporal_block $block
            set text_file [open "$blocktype/$blockname/reports/routeopt/selected/QoR_reformat.csv" "r"]
            echo "$blocktype/$blockname/reports/routeopt/selected/QoR_reformat.csv"
            set x [read $text_file]

            foreach line [split $x "\n"] {
if {$clock_QoR_flag} {
   set clock_QoR_flag 0
   if {[regexp
{^\s*(\S+),\s*(\S+),\s*(\S+),\s*(\S+),\s*(\S+),\s*(\S+),\s*(\S+),\s*(\S+),\s*(\S+),\s*(\S+),\s*(\S+),\s*(\S+),\s*(\S+),\s*(\S+)\
```

```
s*$} $line match wns tns setup_fep clockslew_fep skew latency ccells crepeaters ccells_area
crepeaters_area dyn_clock_power leak_clock_power tot_clock_power]} {
        echo "Data fetched"

        ### Array assignment to populate csv
        set wns_format($blocktype$blockname) $wns
        set tns_format($blocktype$blockname) $tns
        set setup_fep_format($blocktype$blockname) $setup_fep
        set clockslew_fep_format($blocktype$blockname) $clockslew_fep
        set skew_format($blocktype$blockname) $skew
        set latency_format($blocktype$blockname) $latency
        set ccells_format($blocktype$blockname) $ccells
        set crepeaters_format($blocktype$blockname) $crepeaters
        set ccells_area_format($blocktype$blockname) $ccells_area
        set crepeaters_area_format($blocktype$blockname) $crepeaters_area
        set dyn_clock_power_format($blocktype$blockname) $dyn_clock_power
        set leak_clock_power_format($blocktype$blockname) $leak_clock_power
        set tot_clock_power_format($blocktype$blockname) $tot_clock_power

    }
}
if {$general_QoR_flag} {
    if {[regexp {^\s*(\S+),\s*(\S+),\s*(\S+),\s*(\S+),\s*(\S+),\s*(\S+),\s*(\S+)\s*$} $line match utilization tot_area
tot_stdcellarea tot_dyn_power tot_leak_power tot_tot_power DRC_number]} {
        echo "Data fetched"
        echo "$utilization $tot_area $tot_stdcellarea"

        ### Array assignment to populate csv
        set utilization_format($blocktype$blockname) $utilization
        set tot_area_format($blocktype$blockname) $tot_area
        set tot_stdcellarea_format($blocktype$blockname) $tot_stdcellarea
        set tot_dyn_power_format($blocktype$blockname) $tot_dyn_power
        set tot_leak_power_format($blocktype$blockname) $tot_leak_power
        set tot_tot_power_format($blocktype$blockname) $tot_tot_power
        set DRC_number_format($blocktype$blockname) $DRC_number
    }
}
                if {[regexp {^\s*WNS} $line match]} {
                    set clock_QoR_flag 1
                }
                if {[regexp {^\s*Utilization} $line match]} {
                    set general_QoR_flag 1
                }
            }
        }
        set temporal_list {}
    }

    if {$math_flag} {
        echo "Percentage generation compared to the first block given"
foreach blocktype $blocklist {
    if {$blocktype == "block_4" && $block_4_exists} {set temporal_list $list_block_4}
    if {$blocktype == "block_1" && $block_1_exists} {set temporal_list $list_block_1}
    if {$blocktype == "block_2" && $block_2_exists} {set temporal_list $list_block_2}
    if {$blocktype == "block_3" && $block_3_exists} {set temporal_list $list_block_3}

    set reference_initialblock [lindex $temporal_list 0]

    ###Reference values to obtain percentages:

    ###Clock metrics
    set ref_wns $wns_format($blocktype$reference_initialblock)
    set ref_tns $tns_format($blocktype$reference_initialblock)
    set ref_setup_fep $setup_fep_format($blocktype$reference_initialblock)
    set ref_clockslew_fep $clockslew_fep_format($blocktype$reference_initialblock)
    set ref_skew $skew_format($blocktype$reference_initialblock)
```

```
    set ref_latency $latency_format($blocktype$reference_initialblock)
    set ref_ccells $ccells_format($blocktype$reference_initialblock)
    set ref_crepeaters $crepeaters_format($blocktype$reference_initialblock)
    set ref_ccells_area $ccells_area_format($blocktype$reference_initialblock)
    set ref_crepeaters_area $crepeaters_area_format($blocktype$reference_initialblock)
    set ref_dyn_clock_power $dyn_clock_power_format($blocktype$reference_initialblock)
    set ref_leak_clock_power $leak_clock_power_format($blocktype$reference_initialblock)
    set ref_tot_clock_power $tot_clock_power_format($blocktype$reference_initialblock)

    ###General Metrics
    set ref_utilization $utilization_format($blocktype$reference_initialblock)
    set ref_tot_area $tot_area_format($blocktype$reference_initialblock)
    set ref_tot_stdcellarea $tot_stdcellarea_format($blocktype$reference_initialblock)
    set ref_tot_dyn_power $tot_dyn_power_format($blocktype$reference_initialblock)
    set ref_tot_leak_power $tot_leak_power_format($blocktype$reference_initialblock)
    set ref_tot_tot_power $tot_tot_power_format($blocktype$reference_initialblock)
    set ref_DRC_number $DRC_number_format($blocktype$reference_initialblock)



            foreach blockname $temporal_list {
                ### Foreach block not being the reference one:
                ### Exceptions to be contempted:
                    #NaN
                    #ref == 0
                if {$blockname != "$reference_initialblock"} {


                    ###Clock QoR metrics.

                    set int [expr {$wns_format($blocktype$blockname) - $ref_wns}]
                    if {$ref_wns != 0 && $ref_wns != "~"} {set perc_wns($blocktype$blockname) [expr
{$int*100/$ref_wns}]} else {set perc_wns($blocktype$blockname) "NaN"}

                    set int [expr {$tns_format($blocktype$blockname) - $ref_tns}]
                    if {$ref_tns != 0 && $ref_wns != "~"} {set perc_tns($blocktype$blockname) [expr
{$int*100/$ref_tns}]} else { set perc_tns($blocktype$blockname) "NaN"}
                    #echo $perc_tns($blocktype$blockname)
                    echo $blocktype$blockname
                    echo $ref_setup_fep
                    echo $setup_fep_format($blocktype$blockname)

                    if {[string is integer $ref_setup_fep] && [string is integer
$setup_fep_format($blocktype$blockname)]} {set int [expr {$setup_fep_format($blocktype$blockname) -
$ref_setup_fep}]}
                    if {[string is integer $ref_setup_fep] && [string is integer
$setup_fep_format($blocktype$blockname)]} {set perc_setup_fep($blocktype$blockname) [expr
{$int*100/$ref_setup_fep}]} else {set perc_setup_fep($blocktype$blockname) "NaN"}
                    echo $ref_setup_fep
                    if {[string is integer $ref_clockslew_fep] && [string is integer
$clockslew_fep_format($blocktype$blockname)]} {set int [expr
{$clockslew_fep_format($blocktype$blockname) - $ref_clockslew_fep}]}
                    if {[string is integer $ref_clockslew_fep] && [string is integer
$clockslew_fep_format($blocktype$blockname)]} {set perc_clockslew_fep($blocktype$blockname) [expr
{$int*100/$ref_clockslew_fep}]} else {set perc_clockslew_fep($blocktype$blockname) "NaN"}

set int [expr {$skew_format($blocktype$blockname) - $ref_skew}]
if {$ref_skew != 0 && $ref_skew != "~"} {set perc_skew($blocktype$blockname) [expr
{$int*100/$ref_skew}]} else {set perc_skew($blocktype$blockname) "NaN"}

                    set int [expr {$latency_format($blocktype$blockname) - $ref_latency}]
                    if {$ref_latency != 0 && $ref_latency != "~"} {set perc_latency($blocktype$blockname) [expr
{$int*100/$ref_latency}]} else {set perc_latency($blocktype$blockname) "NaN"}
```

```
                set int [expr {$ccells_format($blocktype$blockname) - $ref_ccells}]
                if {$ref_ccells != 0 && $ref_ccells != "~"} {set perc_ccells($blocktype$blockname) [expr
{$int*100/$ref_ccells}]} else {set perc_ccells($blocktype$blockname) "NaN"}

                set int [expr {$crepeaters_format($blocktype$blockname) - $ref_crepeaters}]
                if {$ref_crepeaters != 0 && $ref_crepeaters != "~"} {set
perc_crepeaters($blocktype$blockname) [expr {$int*100/$ref_crepeaters}]} else {set
perc_crepeaters($blocktype$blockname) "NaN"}

                set int [expr {$ccells_area_format($blocktype$blockname) - $ref_ccells_area}]
                if {$ref_ccells_area != 0 && $ref_ccells_area != "~"} {set
perc_ccells_area($blocktype$blockname) [expr {$int*100/$ref_ccells_area}]} else {set
perc_ccells_area($blocktype$blockname) "NaN"}

                set int [expr {$crepeaters_area_format($blocktype$blockname) - $ref_crepeaters_area}]
                if {$ref_crepeaters_area != 0 && $ref_crepeaters_area != "~"} {set
perc_crepeaters_area($blocktype$blockname) [expr {$int*100/$ref_crepeaters_area}]} else {set
perc_crepeaters_area($blocktype$blockname) "NaN"}

                set int [expr {$dyn_clock_power_format($blocktype$blockname) - $ref_dyn_clock_power}]
                if {$ref_dyn_clock_power != 0 && $ref_dyn_clock_power != "~"} {set
perc_dyn_clock_power($blocktype$blockname) [expr {$int*100/$ref_dyn_clock_power}]} else {set
perc_dyn_clock_power($blocktype$blockname) "NaN"}

                set int [expr {$leak_clock_power_format($blocktype$blockname) - $ref_leak_clock_power}]
                if {$ref_leak_clock_power != 0 && $ref_leak_clock_power != "~"} {set
perc_leak_clock_power($blocktype$blockname) [expr {$int*100/$ref_leak_clock_power}]} else {set
perc_leak_clock_power($blocktype$blockname) "NaN"}

                set int [expr {$tot_clock_power_format($blocktype$blockname) - $ref_tot_clock_power}]
                if {$ref_tot_clock_power != 0 && $ref_tot_clock_power != "~"} {set
perc_tot_clock_power($blocktype$blockname) [expr {$int*100/$ref_tot_clock_power}]} else {set
perc_tot_clock_power($blocktype$blockname) "NaN"}

                ### General metrics:

                set int [expr {$utilization_format($blocktype$blockname) - $ref_utilization}]
                if {$ref_utilization!= 0 && $ref_utilization != "~"} {set perc_utilization($blocktype$blockname)
[expr {$int*100/$ref_utilization}]} else {set perc_utilization($blocktype$blockname) "NaN"}

                set int [expr {$tot_area_format($blocktype$blockname) - $ref_tot_area}]
                if {$ref_tot_area != 0 && $ref_tot_area != "~"} {set perc_tot_area($blocktype$blockname)
[expr {$int*100/$ref_tot_area}]} else {set perc_tot_area($blocktype$blockname) "NaN"}

                set int [expr {$tot_stdcellarea_format($blocktype$blockname) - $ref_tot_stdcellarea}]
                if {$ref_tot_stdcellarea != 0 && $ref_tot_stdcellarea != "~"} {set
perc_tot_stdcellarea($blocktype$blockname) [expr {$int*100/$ref_tot_stdcellarea}]} else {set
perc_tot_stdcellarea($blocktype$blockname) "NaN"}

                set int [expr {$tot_dyn_power_format($blocktype$blockname) - $ref_tot_dyn_power}]
                if {$ref_tot_dyn_power != 0 && $ref_tot_dyn_power != "~"} {set
perc_tot_dyn_power($blocktype$blockname) [expr {$int*100/$ref_tot_dyn_power}]} else {set
perc_tot_dyn_power($blocktype$blockname) "NaN"}

                set int [expr {$tot_leak_power_format($blocktype$blockname) - $ref_tot_leak_power}]
                if {$ref_tot_leak_power != 0 && $ref_tot_leak_power != "~"} {set
perc_tot_leak_power($blocktype$blockname) [expr {$int*100/$ref_tot_leak_power}]} else {set
perc_tot_leak_power($blocktype$blockname) "NaN"}

                set int [expr {$tot_tot_power_format($blocktype$blockname) - $ref_tot_tot_power}]
                if {$ref_tot_tot_power != 0 && $ref_tot_tot_power != "~"} {set
perc_tot_tot_power($blocktype$blockname) [expr {$int*100/$ref_tot_tot_power}]} else {set
perc_tot_tot_power($blocktype$blockname) "NaN"}

                set int [expr {$DRC_number_format($blocktype$blockname) - $ref_DRC_number}]
```

```
                if {$ref_DRC_number != 0 && $ref_DRC_number != "~"} {set
perc_DRC_number($blocktype$blockname) [expr {$int*100/$ref_DRC_number}]} else {set
perc_DRC_number($blocktype$blockname) "NaN"}

            }
        }
        unset ref_wns
        unset ref_tns
        unset ref_setup_fep
        unset ref_clockslew_fep
    }
}
if {$multi_files_flag} {
    echo "Each block will have its own .csv file"
    foreach blocktype $blocklist {
        ### Assigning temporal lists to manage less code for iterations:

        if {$blocktype == "block_4" && $block_4_exists} {set temporal_list $list_block_4}
        if {$blocktype == "Block_1" && $Block_1_exists} {set temporal_list $list_Block_1}
        if {$blocktype == "Block_2" && $Block_2_exists} {set temporal_list $list_Block_2}
        if {$blocktype == "block_3" && $block_3_exists} {set temporal_list $list_block_3}

        set reference_initialblock [lindex $temporal_list 0]

        ### Opening of individual csv file:

        set csv_file "csv_block_combiner/${blocktype}.csv"
        set csv [open $csv_file "w"]
        puts $csv "QoR for $blocktype"
        puts $csv "Clock QoR metrics\n"
        puts $csv $clock_QoR_baseline
        foreach blockname $temporal_list {
```

```
            puts $csv "$blockname, $wns_format($blocktype$blockname),
$tns_format($blocktype$blockname), $setup_fep_format($blocktype$blockname),
$clockslew_fep_format($blocktype$blockname), $skew_format($blocktype$blockname),
$latency_format($blocktype$blockname), $ccells_format($blocktype$blockname),
$crepeaters_format($blocktype$blockname), $ccells_area_format($blocktype$blockname),
$crepeaters_area_format($blocktype$blockname),
$dyn_clock_power_format($blocktype$blockname),
$leak_clock_power_format($blocktype$blockname),
$tot_clock_power_format($blocktype$blockname)"
        }
        puts $csv "\n"
        puts $csv "General QoR metrics \n"
        puts $csv $general_QoR_baseline
        foreach blockname $temporal_list {
            puts $csv "$blockname, $utilization_format($blocktype$blockname),
$tot_area_format($blocktype$blockname), $tot_stdcellarea_format($blocktype$blockname),
$tot_dyn_power_format($blocktype$blockname), $tot_leak_power_format($blocktype$blockname),
$tot_tot_power_format($blocktype$blockname), $DRC_number_format($blocktype$blockname)"
        }
        if {$math_flag} {
            puts $csv "\n"
            puts $csv "Clock QoR percentage metrics: value = (current_value -
reference_value)*100/reference_value"
            puts $csv $clock_QoR_baseline
            foreach blockname $temporal_list {
                if {$blockname != $reference_initialblock} {
```

```
                    puts $csv "$blockname, $perc_wns($blocktype$blockname),
$perc_tns($blocktype$blockname), $perc_setup_fep($blocktype$blockname),
$perc_clockslew_fep($blocktype$blockname),$perc_skew($blocktype$blockname),
$perc_latency($blocktype$blockname), $perc_ccells($blocktype$blockname),
$perc_crepeaters($blocktype$blockname), $perc_ccells_area($blocktype$blockname),
$perc_crepeaters_area($blocktype$blockname), $perc_dyn_clock_power($blocktype$blockname),
$perc_leak_clock_power($blocktype$blockname), $perc_tot_clock_power($blocktype$blockname)"
                  }
               }
               puts $csv "\n"
               puts $csv "General QoR percentage metrics: value = (current_value -
reference_value)*100/reference_value"
               puts $csv $general_QoR_baseline
               foreach blockname $temporal_list {
                  if {$blockname != $reference_initialblock} {
                     puts $csv "$blockname, $perc_utilization($blocktype$blockname),
$perc_tot_area($blocktype$blockname), $perc_tot_stdcellarea($blocktype$blockname),
$perc_tot_dyn_power($blocktype$blockname), $perc_tot_leak_power($blocktype$blockname),
$perc_tot_tot_power($blocktype$blockname), $perc_DRC_number($blocktype$blockname)"
                  }
               }
            }
         close $csv
         }
      } else {
         echo "A single csv file will be generated for all blocks\n"

         set csv_file "csv_block_combiner/merged_QoR.csv"
         set csv [open $csv_file "w"]

         foreach blocktype $blocklist {

            ### Assigning temporal lists to manage less code for iterations:

            if {$blocktype == "block_4" && $block_4_exists} {set temporal_list $list_block_4}
            if {$blocktype == "Block_1" && $Block_1_exists} {set temporal_list $list_Block_1}
            if {$blocktype == "Block_2" && $Block_2_exists} {set temporal_list $list_Block_2}
            if {$blocktype == "block_3" && $block_3_exists} {set temporal_list $list_block_3}

            set reference_initialblock [lindex $temporal_list 0]
            puts $csv "\n"
            puts $csv "QoR for $blocktype"
            puts $csv "Clock QoR metrics\n"
            puts $csv $clock_QoR_baseline

            foreach blockname $temporal_list {
               puts $csv "$blockname, $wns_format($blocktype$blockname),
$tns_format($blocktype$blockname), $setup_fep_format($blocktype$blockname),
$clockslew_fep_format($blocktype$blockname), $skew_format($blocktype$blockname),
$latency_format($blocktype$blockname), $ccells_format($blocktype$blockname),
$crepeaters_format($blocktype$blockname), $ccells_area_format($blocktype$blockname),
$crepeaters_area_format($blocktype$blockname),
$dyn_clock_power_format($blocktype$blockname),
$leak_clock_power_format($blocktype$blockname),
$tot_clock_power_format($blocktype$blockname)"
            }
            puts $csv "\n"
            puts $csv "General QoR metrics \n"
            puts $csv $general_QoR_baseline

            foreach blockname $temporal_list {
               puts $csv "$blockname, $utilization_format($blocktype$blockname),
$tot_area_format($blocktype$blockname), $tot_stdcellarea_format($blocktype$blockname),
$tot_dyn_power_format($blocktype$blockname), $tot_leak_power_format($blocktype$blockname),
$tot_tot_power_format($blocktype$blockname), $DRC_number_format($blocktype$blockname)"
            }
```

```
            if {$math_flag} {
                puts $csv "\n"
                puts $csv "Clock QoR percentage metrics: value = (current_value -
reference_value)*100/reference_value"
                puts $csv $clock_QoR_baseline

                foreach blockname $temporal_list {
                    if {$blockname != $reference_initialblock} {
                        puts $csv "$blockname, $perc_wns($blocktype$blockname),
$perc_tns($blocktype$blockname), $perc_setup_fep($blocktype$blockname),
$perc_clockslew_fep($blocktype$blockname), $perc_skew($blocktype$blockname),
$perc_latency($blocktype$blockname), $perc_ccells($blocktype$blockname),
$perc_crepeaters($blocktype$blockname), $perc_ccells_area($blocktype$blockname),
$perc_crepeaters_area($blocktype$blockname), $perc_dyn_clock_power($blocktype$blockname),
$perc_leak_clock_power($blocktype$blockname), $perc_tot_clock_power($blocktype$blockname)"
                    }
                }
                puts $csv "\n"
                puts $csv "General QoR percentage metrics: value = (current_value -
reference_value)*100/reference_value"
                puts $csv $general_QoR_baseline

                foreach blockname $temporal_list {
                    if {$blockname != $reference_initialblock} {
                        puts $csv "$blockname, $perc_utilization($blocktype$blockname),
$perc_tot_area($blocktype$blockname), $perc_tot_stdcellarea($blocktype$blockname),
$perc_tot_dyn_power($blocktype$blockname), $perc_tot_leak_power($blocktype$blockname),
$perc_tot_tot_power($blocktype$blockname), $perc_DRC_number($blocktype$blockname)"
                    }
                }
            }

        }
        close $csv
    }
}

define_proc_attributes block_combiner -info "Merges existing reports into a single file" \
   -define_args {
      {-tee "Optional - displays the output of under-the-hood clock_structure_analyzer command" ""
boolean optional}
      {-block_4_list "Optional - Passes the name of the block_4 block reports to be merged" "" list
optional}
      {-block_1_list "Optional - Passes the name of the Block_1 block reports to be merged" "" list
optional}
      {-block_2_list "Optional - Passes the name of the Block_2 block reports to be merged" "" list
optional}
      {-separate_files "Optional - Specifies if one file per block is used on report_generation" "" boolean
optional}
      {-math "Optional - Specifies if percentages on values compared to the first block on each list is
made" "" boolean optional}
      {-block_3_list "Optional - Passes the name of the block_3 block reports to be merged" "" list
optional}
   }
echo "\tblock_combiner"
```

Glossary

ADB: Adjustable Delay Buffer

CCD: Concurrent Clock and Data Optimization

CSV: Comma Separated Values

DRC: Design Rule Check

EDA: Electronic Design Automation

GUI: Graphical User Interface

ICG: Integrated Clock Gating Cell

MCMM: Multi-Corner Multi-Mode

OCV: On-Chip Variation

PPA: Power Performance Analysis

PVT: Power-Voltage-Temperature

QoR: Quality of Results

TCL: Tool Command Language

TNS: Total Negative Slack

WNS: Worst Negative Slack